

Mobile Phone based Remote Monitoring System

by

Danyi Liu

**A Thesis Submitted in Fulfilment of
the Degree of
Master of Engineering**



**Auckland University of Technology
Auckland**

June 2008

Acknowledgements

I wish to express my gratitude to my supervisor, Professor Adnan Al-Anbuky, for his ongoing guidance, his patience, his excellent advice, and most of all his kind understanding. His high expectations of me encouraged me to perform the best that I could and I respect him for that.

I would also like to express my gratitude to Dr. Lin Chen, who had been my co-supervisor for one semester, for his patience and his good advice.

I would also like to thank Murray McGovern at Mobile Control Solutions Ltd for his technical and project support. His help has been very valuable.

Thanks also to Mr. Hong Zhang at MCS and to Sean Tindle, both of whom tolerated me and aided me in my quest so supportively.

Thanks to David Parker for his proof reading of my work and his advice.

I appreciate also, so much, my family's interest and encouragement, without which I would not have had this opportunity.

Abstract

This thesis investigates embedded databases and graphical interfaces for the MicroBaseJ project. The project aim is the development of an integrated database and GUI user interface for a typical 3G, or 2.5G, mobile phone with Java MIDP2 capability. This includes methods for data acquisition, mobile data and information communication, data management, and remote user interface. Support of phone delivered informatics will require integrated server and networking infrastructure research and development to support effective and timely delivery of data for incorporation in mobile device-based informatics applications. A key research and development (R&D) challenge is to support effective and timely delivery of data for incorporation in mobile device-based informatics applications. Another important aspect of the project is determining how to develop efficient graphics for the small mobile screen.

The research investigates and analyses the architecture of a mobile monitoring system. The project developed a generic solution that can be implemented in a number of commercial sectors, such as horticulture, building management and pollution/water management. The developed concept is tested using data relevant to the horticultural area of application. The system also addresses the main issues related to mobile monitoring, including realtime response, data integrity, solution cost, graphical presentation, and persistent storage capabilities of modern mobile devices.

Four embedded databases based on J2ME have been investigated. Two of the four have been evaluated and analysed. The Insert function, Sequence Search, and Random Search of Perst List and RMS (Record Management System) databases have been tested. The size of the processed data was limited to 20,000 records when using the wireless toolkit simulator, and 11,000 records when using a mobile phone. Perst Lite reflects good performance and has out-performed RMS in all tests.

User interface software such as J2ME Polish for mobile phones has been investigated. Custom J2ME class for graphical interface is developed. This provides the graphical presentation of the data collected from the sensors; including temperature, wind speed,

wind direction, moisture, and leaf wetness. The graphical interface, bar charts, and line charts with trace ball for collected data have been designed and implemented.

The embedded database performance and project performance have been investigated and analysed. The performances of Perst Lite and RMS are evaluated in terms of the insert, sequence search, and random search functions based on simulation and real devices. The record numbers vary from 1,000 to 20,000. The project performance contains data receiving and storage, and data presentation and configuration. The performance of data storage and configuration can be negated due to the running mode and the response time. Thus, data presenting performance is the key focus in this project. This performance was divided into the categories of initial, data search, data selection, and charting. The initial performance includes the initialisation of the project parameters, and the reaching of the welcome interface. Data search performance refers to the retrieval of the specified data from the embedded database, measured on 48 data points, which only can be presented on the mobile screen from the retrieved data. These four performance types are measured in thousands of record numbers, varying from 1,000 to 18,000 record numbers, with the retrieved data range varying from 1 day to 30 days.

Table of contents

Acknowledgements	i
Abstract	ii
Table of contents	iv
List of figures	x
List of tables	xii
List of tables	xii
List of abbreviations.....	xiii
Statement of originality.....	xvii
1 Introduction	1
1.1 Introduction	1
1.2 The Project Background.....	1
1.3 Literature Review.....	2
1.3.1 Mobile Application Field and Architecture	2
1.3.1.1 Control Applications	2
1.3.1.2 Positional Application.....	3
1.3.1.3 Telemedicine Application	4
1.3.1.4 Education Application.....	4
1.3.2 Mobile Database Applications	5

1.3.3	Mobile Application Interface Review	6
1.3.4	Wireless Communication Protocol	7
1.3.4.1	SMS.....	7
1.3.4.2	Bluetooth.....	7
1.3.4.3	WAP.....	8
1.3.5	Mobile Development Platform.....	8
1.3.5.1	J2ME	8
1.3.5.2	Binary Runtime Environment for Wireless (BREW)	10
1.3.5.3	Windows Electric Compact (Windows CE)	10
1.4	Embedded Databases	11
1.4.1	Perst Lite	11
1.4.2	RMS	12
1.4.3	Pointbase	12
1.4.4	db4o.....	12
1.5	Mobile User Interface	14
1.5.1	Human-machine Communication	15
1.5.2	Psychological Considerations	15
1.5.3	Hardware for User Interface Communication.....	16
1.5.4	Interface Design Principles	17
1.5.4.1	General Principles	17
1.5.4.2	Parameter Description Syntax.....	17

1.5.4.3	Screen Layouts	18
1.5.4.4	Commands	19
1.5.4.5	Menus	20
1.6	Project Objective	20
1.7	Project Plan	21
2	Development Environment and Tools	23
2.1	Introduction	23
2.2	Environment	23
2.2.1	Development Environment	23
2.2.2	Mobile handset	24
2.2.3	Application Platform	25
2.2.3.1	TINI	26
2.2.3.2	Wavecom Fastrack	26
2.3	Tools	27
2.3.1	IDE environment	27
2.3.2	Software	27
2.3.2.1	J2ME	28
2.3.2.2	J2ME Polish	29
2.4	Summary	30
3	System Design	31
3.1	Introduction	31

3.2	Requirement Analysis	31
3.2.1	User Requirement	31
3.2.2	Data Sources	32
3.2.3	The Proposed Structure for MicroBaseJ	34
3.2.4	Communication Design.....	35
3.2.5	Functional Requirement	35
3.2.6	Non-Functional Requirement.....	37
3.2.7	MicroBaseJ Roadmap	38
3.2.8	Interface Description.....	39
3.2.9	Data Structure between Components	40
3.2.10	Dataflow Model	42
3.3	System Design.....	44
3.3.1	Class Diagram	45
3.3.2	State Diagram.....	49
3.4	Summary	53
4	Mobile Phone Database and User Interface Implementation.....	54
4.1	Introduction	54
4.2	Database Design.....	54
4.2.1	Data Format.....	54
4.2.2	Data Storage	55
4.3	User Interface Design.....	57

4.3.1	Hardware for User Interface.....	58
4.3.1.1	Screen Size	58
4.3.1.2	Other devices.....	59
4.3.2	Software Chosen for User Interface.....	59
4.3.3	User Interface Design.....	60
4.3.3.1	Screen Layouts	61
4.3.3.2	Commands and Menus.....	62
4.3.3.3	The Design for the Graphic User Interface	63
4.4	Summary	66
5	Evaluation and Discussion	67
5.1	Introduction	67
5.2	Database Performance Evaluation (Perst Lite & RMS).....	67
5.2.1	Simulation	68
5.2.1.1	Insert Function	68
5.2.1.2	Sequence Search Function	69
5.2.1.3	Random Search Function	70
5.2.1.4	Database Size Comparison.....	72
5.2.2	Real Device	73
5.2.2.1	Insert Function	73
5.2.2.2	Sequence Search Function	74
5.2.2.3	Random Search Function	75

5.2.2.4	JAR Size Comparison	76
5.2.3	Comparison on 5,000 Records	77
5.2.4	Database Discussion	78
5.3	Project Performance Evaluation.....	79
5.3.1	Analysis of the Project	79
5.3.2	Initialisation Performance	81
5.3.3	Data Search Performance	83
5.3.4	Data Selection Performance	84
5.3.5	Data Plotting	85
5.3.6	Discussion of the Project.....	86
6	Conclusion	88
6.1	Introduction	88
6.2	Conclusion	88
6.3	Future Work	90
	References	91
	Appendix I – Sample Mobile Phones and their Resolutions.....	96
	Appendix II – The Introduction of the User Interfaces.....	98
	Appendix III – The Conference Paper for ATNAC’07	107
	Appendix IV – The CD contents.....	114

List of figures

Figure 1: The Java Platform.....	9
Figure 2: CLDC Wireless Platform	9
Figure 3: The Development Environment for MicroBaseJ.....	24
Figure 4: The MicroBaseJ Platform.....	26
Figure 5: The architecture of MicroBaseJ.....	34
Figure 6: The Interface Diagram for MicroBaseJ	40
Figure 7: The Dataflow of MicroBaseJ.....	43
Figure 8: The Class Diagram for MicroBaseJ.....	45
Figure 9: The Illustration for the ReceiveSMSMsg Class	47
Figure 10: The Illustration for the MainMenu class	48
Figure 11: The Illustration for the myCanvas Class	49
Figure 12: The Statechart Diagram for the ReceiveSMSMsg class.....	50
Figure 13: Statechart Diagram for the MainMenu class	52
Figure 14: The minimum resolution for mobile phones	58
Figure 15: The Mobile User Interface Design for Chart.....	64
Figure 16: The Illustration of the Graphic Interface	66
Figure 17: The Insert Function Comparison on Simulation	69
Figure 18: The Sequence Search Comparison of Perst Lite and RMS	70
Figure 19: Random Search Comparison for Perst Lite with and without index and RMS	71

Figure 20: The Insert Function Comparison of Perst Lite and RMS on Mobile	73
Figure 21: The Random Search Comparison of Perst Lite and RMS on Mobile	74
Figure 22: The Random Search Comparison of Perst Lite with Index and without Index and RMS on mobile	75
Figure 23: The Initialisation Performance Comparison.....	82
Figure 24: The Data Search Performance Comparison.....	83
Figure 25: The Data Selection Performance	85
Figure 26: The Data Plotting Performance Comparison.....	86
Figure 27 - The Splash Interface.....	98
Figure 28 - The Main Menu	99
Figure 29 - The Item Choice Menu for Chart	100
Figure 30 - The Date and Period Entry Menu.....	101
Figure 31 - The Three types of Chart.....	102
Figure 32 – The Alarm Item Choice Menu.....	103
Figure 33 - The Modified Alarm Menu	104
Figure 34 - Confirmation for Updated Alarm.....	105
Figure 35 - Sample Rate Modified Menu	105
Figure 36 - Confirmation for Sample Rate Updating.....	106

List of tables

Table 1: Database Comparison	14
Table 2 - The Schedule for MicroBaseJ.....	21
Table 3: The relation of values and wind direction.....	32
Table 4: The functional requirements for MicroBaseJ	36
Table 5: The Non-Functional Requirements for MicroBaseJ	37
Table 6: MicroBaseJ Roadmap	38
Table 7: Data Format for Messages from Sentinel to Mobile Phone.....	41
Table 8: Sample Rate Format for Configuration to the Sentinel	42
Table 9 - Database format for MicroBaseJ	55
Table 10 - The size comparison of RMS & Perst Lite.....	56
Table 11: The Relation of Main Menu Items and Icons	61
Table 12 - The Relation of Data Source and Icon.....	61
Table 13: The Database Size Comparison of Perst Lite and RMS	72
Table 14: The Jar File Size Comparison for Perst Lite with Index and without Index and RMS	76
Table 15: The Comparison of Perst Lite and RMS on the Simulation and the Mobile at 5,000 Records.....	77
Table 16: The Maximum of the Random Retrieved Date for Data Search.....	81

List of abbreviations

3D	Three dimensional
3G	The third generation
ACID	Atomicity, Consistency, Isolation, Durability
API	Application programming interface
BREW	Binary Runtime Environment for Wireless
CCU	Critical care unit
CDC	Connected Device Configuration
CDMA	Code Division Multiple Access
CLA	Compass location adapter
CLDC	Connected Limited Device Configuration
CT	Computerised tomographic
DBMS	Database management system
DSK	Development Software Kit
DTV	Digital TV
EDGE	Enhanced data rates for GSM evolution
EGSM	Extension of GSM
GIS	Geographic information system
GPS	Global Positioning System

GPRS	General Packet Radio Service
GSM	Global System for Mobile communication
GUI	Graphical User Interface
HCI	Human-computer interaction
HTML	HyperText Markup Language
ICU	Intensive care unit
IDE	Integrated development environment
I/O	Input/output
JAR	Java Archive
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JDK	Java Development Kit
IRE	Information Requirement Elicitation
JSR	Java Specification Request
JTWI	Java Technology for the Wireless Industry
JVM	Java virtual machine
MCS	Mobile Control Solutions Ltd.
MIDP	Mobile Information Device Profile
MR	Magnetic resonance
MSA	Mobile Service Architecture

OODBMS	Object oriented database management system
OS	Operating system
ORDBMS	Object relational database management system
OTA	Over-the-air
PAN	Personal area network
PC	Personal computer
PDA	Personal digital assistant
PIM	Personal information management
PIN	Personal identification number
POP3	Post Office Protocol version 3
QAR	Question-Answer Relationship
QBE	Query-By-Example
QVGA	Quarter Video Graphics Array
R&D	Research and development
RDBMS	Relational Database Management System
RMS	Record Management System
UML	Unified Modelling Language
UMTS	Universal Mobile Telecommunication
US	Ultrasonography
VCR	Video Cassette Recorder
WAP	Wireless Access Protocol

WCDMA	Wideband Code Division Multiple Access
WEHP	WAP enabled handphone
WEB	A computer programming system
WISMO	Wireless Standard Module
WMAPI	Wireless messaging API
WML	Wireless Markup Language
SD	Secure Digital
SDK	Software development kit
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SODIMM	Small outline dual in-line memory module
TFT	Thin-film transistor
XML	Extensible Markup Language

Statement of originality

‘I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the qualification of any other degree or diploma of a university or other institution of higher learning, except where due acknowledgement is made in the acknowledgements.’

Danyi Liu

(signed)

----- (date)

1 Introduction

1.1 Introduction

The thesis proposes a remotely controlled mobile application for horticulture. This chapter introduces the project's background and reviews mobile application areas, databases, and communication protocols for mobile applications. It also discusses some embedded databases available for mobile applications. Some topics for user interface theory and design are presented in this chapter. Next, the project objective and project plan are proposed.

1.2 The Project Background

Mobile phones have been part of our lives for over a decade. There have been many research studies on the use of wireless services for remote monitoring and control over these years [1]. In the early stage of mobile phone usage, however, the growth of mobile applications had been limited by the non-availability of efficient handsets and secure wireless networks.

With a global mobile phone user base in excess of 1.3 billion and an acknowledged need for current information-based management processes it is believed that there is a current market base for at least 75,000 applications in the Australasian market in the horticultural and water/irrigation management sectors. Further generic opportunities exist in chilled assets, industry, energy, pollution, and security related applications. Bass [2] product adoption models suggest a technology uptake over five to seven years in this technology sector. In recognition of the importance of the project to the industry, it has been coordinated by MCS (Mobile Control Solutions) Ltd., a lead developer who has industry experience and current high level academic contacts in the mobile device applications development sector.

1.3 Literature Review

1.3.1 Mobile Application Field and Architecture

There have been a number of research projects related to the use of the cell phone as a remote monitor and controller. Most of these projects have focused on telemedicine, education, control of plant and home appliances, and spatial information services. Below we discuss concisely some of the popular application areas.

1.3.1.1 Control Applications

Home appliance control is the most popular field for mobile application. Nikolova, Meijs and Voorwinden [3] developed a technique for interconnecting home and mobile networks to enable the control of home appliances from a remote mobile phone, or a web pad. The remote control functions include remote mobile programming of VCRs (video cassette recorders), remote mobile control of heating thermostats, and remote mobile monitoring using security cameras. Another control system for home appliances was presented by Nichols and Myers [4]. This system can automatically generate interfaces with appliances from abstract specifications of the functions of the appliances, with the interfaces allowing users to control all functions of the appliances from their smart phones. Moreover, Ishikawa, Saito and Cohen [5] introduced a framework to synchronise avatars and appliances with mobile phone ringtones. The architecture provided an interface to control home appliances and avatars ¹.

Besides common home appliances, set-top-boxes are also involved in control areas. Lin and Chen [6] developed a framework to let users control set-top-boxes from mobile devices, such as mobile phones and laptops. The application allowed mobile users to watch digital TV (DTV) content online and remotely command the set-top-box to record a DTV program. Furthermore, Sirskanthan, Meher, Ng and Heng [7] devised a Teletext WAP access system. The system transformed the Teletext contents from the screen format of a TV to the screen format of a WAP

¹ The 3D CG characters which reflect a client's action.

(wireless application protocol) Enabled Handphone (WEHP), and also allowed WAP enabled mobile phones to access the Teletext-WAP database and show database information on their screens.

Plant control is another area investigated in mobile applications. Ravi, Chathish and Prasanna [1] proposed technical and maintenance personnel supervision and control of machinery and processes from a cellular phone. They used the WAP protocol to develop an alarm management program for providing alert signals when any received data exceeds a preset value for the selected process variables. In addition, a remote monitoring and inspection system for robotic manufacturing was presented by Pires [8]. This system uses Simple Mail Transfer Protocol (SMTP) and Post Office Protocol version 3 (POP3) to transmit warnings and reports to users' mobile phones and beepers.

1.3.1.2 Positional Application

The positional application is another popular area in mobile applications. A user can request specific positional information from a mobile phone, or a PDA (Personal Digital Assistant) with a particular device. Shimada, Tanizaki and Maruyama [9] presented a structure for providing different spatial information services. A compass location adapter (CLA) was developed to assess location and direction into the mobile phone instead of sensor devices, and was mounted on the cell phone. The mobile user submitted requests for spatial information and the particular message would be shown on the screen.

A user cannot only retrieve information with CLA devices, but can also request spatial information from a server, or other device. Rahman and Bhalla [10] presented an interface for spatial data queries on mobile devices. They created extensions for Query-By-Example (QBE), so that the interface could support spatial queries on portable devices. Some positional applications provide different information in terms of the types of mobile devices. A framework for querying hyperlinked multimedia cultural inheritance datasets, such as museum photos, was proposed by Carswell, Eustace, Gardiner, Kilfeather and Neumann [9]. This framework could provide different information to users based on their devices'

categories, such as a mobile phone not receiving image data, and a PDA not receiving video data.

1.3.1.3 Telemedicine Application

There have been many mobile applications targeting the telemedicine field. Some applications allow doctors to access the medical records stored in a remote server. An implementation of a WAP-based telemedicine system was developed by Hung and Zhang [11]. They utilised WAP devices as mobile access PCs for common inquiries and patients' common data. Authorised users could view a patient's data, monitor blood pressure and electrocardiogram results on WAP equipment in store-and-forward mode. At the same time, Andrade, Wangenheim, Bortoluzzi and De Biasi [12] investigated an approach to allow medical staff to access patient records, such as computerised tomographic (CT), ultrasonography (US), and magnetic resonance (MR) images, when visiting the patient's bedside, or in emergency situations. Koop and Mosges [13] also used mobile devices to store patient diaries to increase the quality of data and reduce the time needed to close the database.

Telemedicine applications not only allow doctors to access relevant records, but can also provide monitoring of emergent patients. Kogue, Matsuoka, Kinouchi and Akutagawa [14] proposed a remote patient monitoring system. The system was designed for use with a 3G mobile phone to observe information of various patients in an ICU (Intensive Care Unit), or a CCU (Critical Care Unit). Another application which helped the elderly with dementia was outlined by Lin, Chiu, Hsiao, Lee and Tsai [15]. They developed a platform, including a web service, database, message controller, and health geographic information system (GIS) server, to implement various monitoring schemes, such as indoor residence monitoring and emergency rescue.

1.3.1.4 Education Application

Education is another hot issue in mobile applications. Ketamo [16] introduced an adaptable mobile working environment, xTask, for teaching and training. PC, or PDA, users could access this environment, which was built over software platforms such as Apache web-server, ActivePerl programming interface and

MySQL database. In addition, a framework was introduced for collaborative mobile learning by Black and Hawkes [17]. The framework provided an interface for PDA users to allow them to carry out reading-comprehension testing using Question-Answer Relationship (QAR).

Moreover, databases have also been introduced into mobile education applications. A communication and discussion toolkit, based on sending short messages, designed for use in schools was presented by Bollen, Eimler and Hoppe [18]. The messages generated by the students were collected into a database and then established a basis for discussion and analysis. Meurant [19] reviewed an application to use cell phones in L2 (second language) classrooms. This application captured SMS (Short Message Service) into the database, which was later incorporated into the display on the message board.

Although there are many fields covered in the literature regarding mobile applications, such as home appliance and plant control, spatial query, telemedicine, and education, there are none targeting horticulture.

1.3.2 Mobile Database Applications

Many different types of databases have been applied in a variety of mobile applications. Some database engines are based on the file system. A mobile Web Service system, which used VS.net tools, C# language, and the Asp.net technique, was proposed by Gao, Wang, Jiang and Sun [20]. In addition, Bakos, Farkas and Nurminen [21] proposed a search method for phonebook-based smart phone networks. This search engine allowed users to search the information in ways that were closer to a universal human perception of value and reliability.

Some relational and object databases are also introduced. Rahman and Bhalla [10] proposed an interface that could support queries for spatial data on mobile devices. This interface was designed for a Relational Database Management System (RDBMS). In addition, Rahman, Bhalla and Hashimoto [22] developed a Query-By-Object interface for Information Requirement Elicitation (IRE). This application was built on an Object-Relational Database Management System (ORDBMS). Furthermore, Lo, Chang, Frieder and Grossman [23] compared the dynamics of the performance of eight Java programs; jess, javac, mtrt, compress, db, db4o, smallDB, and ozone; in terms of object

size distribution, average object size, object live distribution, and the total garbage collection cycle. Their results reveal that db, ozone, db4o, and smallDB share several similarities related to the object size, the object live span, and the object size set.

Some real databases used by the above applications are installed in the server side, while some installed in mobile phones are based on a file system. Most embedded databases are designed for PDA, or mobile laptop, with only a few being suitable for cell phones, due to the resource-constraints.

1.3.3 Mobile Application Interface Review

The user interface is the end point for the user. Hence its appropriate design is very important to the user. There have been a number of mobile applications focusing on user interface design over recent years. Imai, Ooga, Yamane, Sadayuki, Iwamoto and Masuda [24] investigated a monitoring system integrating network cameras, an integrated web/mail/database server, and web-based high performance mobile phones. The mobile phone user could access the relevant information located in a server via a graphic user interface. In addition, Yang and Kou [25] proposed a model for monitoring and control of PC clusters from a mobile phone. The user interface for mobile phones was designed in a graphic mode. Further, Rahman, Bhalla and Hashimoto [26] proposed a high level user interface for Information Requirement Elicitation (IRE). A mobile web user can access the server information through the Query-By-Object approach.

Some applications provided a generator for the user interface. Howard and Bradford [27] presented an universal interface generator for a PDA, or mobile phone to control various devices. Mayyora-Ibarra, Paz-Arroyo, Cambranes-Martinez and Fuentes-Penna [28] proposed a tool for designing common user interfaces which could be transcoded to multiple target languages, such as VoiceXML (Extensible Markup Language), J2ME (Java 2 Micro Edition), HTML (HyperText Markup Language), and WML (Wireless Markup Language).

Most research focused on generating a generic control interface for the mobile user, rather than figures or pictures for the interface. Other researches were keen to develop a user interface to retrieve figures or pictures from a server, rather than generating them from the mobile phone itself.

1.3.4 Wireless Communication Protocol

There are many techniques for communication; including Bluetooth, WAP, and SMS; that relate to the distance between the client and server.

1.3.4.1 SMS

Short Message Service (SMS) is a communications protocol which allows the exchange of short text messages between mobile devices. The development and growth of SMS has been significant since it was first developed. There are many applications based on the SMS.

Bollen, Eimler and Hoppe [29] explored a communication and discussion toolkit for use in schools. The environment imitated the sending of short messages using mobile handsets, and stored the message in a database for later discussion and analysis. The application presented by Meurant [19] also captured SMSs into a database in the L2 classroom. These SMS could be displayed on a message board.

1.3.4.2 Bluetooth

Bluetooth is another important protocol for mobile applications. Bluetooth, an industrial specification for wireless personal area networks (PANs), is built and licensed by the Bluetooth Special Interest Group. Devices, such as mobile phones, laptops, personal computers, printers, GPS (Global Positioning System) receivers, digital cameras, and video game consoles, can use Bluetooth over a secure, globally unlicensed short-range radio frequency to connect and swap information [30].

Alhakim, Al-Kittani, Swidan and Zarka [31] presented an infrastructure for remote control of PCs via a mobile handset through Bluetooth. Users utilised mobile phones to access PCs' Internet Explorer, media player, PowerPoint, and MSN messenger programs. An internet home control structure was also investigated by Tan and Soh [32]. This architecture allowed subscribers to use a mobile phone to control home appliances. The protocols used between mobile and server, and between server and home appliances, are WAP and Bluetooth, respectively.

1.3.4.3 WAP

WAP is an open international specification that allows wireless devices such as mobile phones, or PDAs, to access the Internet [33]. This protocol has become more and more important because it enables mobile devices to access the Internet at anytime and from anywhere. An integrated system based on the WEB/WAP framework for remote monitoring and control of industrial processes was proposed by Nikolakopoulos, Koundourakis and Tzes [34]. A user could access the system using a WEB browser, or a WAP-enabled mobile phone.

SMS, Bluetooth and WAP has its advantages and disadvantages. Bluetooth costs nothing to use, but is limited by a transmission distance of up to 100 metres. In comparison, using SMS, a particular text or message can be sent to anyone anywhere, however, there is a cost for using the SMS. Finally, WAP costs the most to use, but it can not only send text and multimedia messages, but can also receive relevant information from the Internet through a mobile phone.

1.3.5 Mobile Development Platform

Many different platforms have been used for the development of mobile applications, such as J2ME, BREW (Binary Runtime Environment for Wireless), and Windows CE.

1.3.5.1 J2ME

J2ME is one of the three Java platforms defined by Sun Microsystems. The others are J2SE (Java 2 Standard Edition) and J2EE (Java 2 Enterprise Edition). J2ME is a flexible and robust environment that could be suitable for mobile devices. It contains a compilation of technologies and specifications for a Java runtime environment, designed to fit the particular device's requirements [35]. J2ME contains stretchy user interfaces, robust security, built-in network protocols, and support for networked and offline applications that can be downloaded dynamically. J2ME consists of configuration, profile, and an optional package [36]. Figure 1 describes an outline of Java ME technology's components and its relation with the other Java technologies [37].

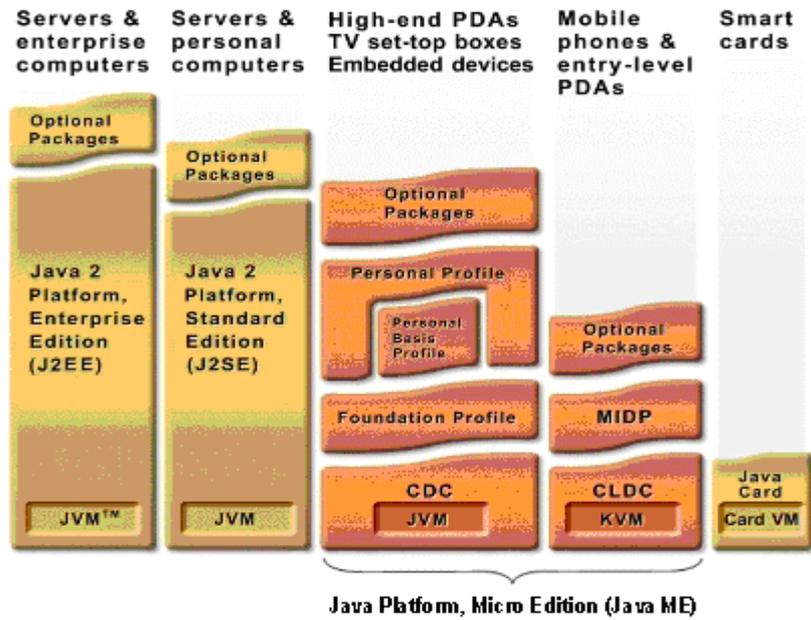


Figure 1: The Java Platform

Figure 2 presents the platform for Connected Limited Device Configuration (CLDC). CLDC targets resource-constraint devices such as mobile phones. The combination of CLDC and the Mobile Information Device Profile (MIDP) can provide a Java application environment for mobile handsets and other devices with similar capabilities [37].

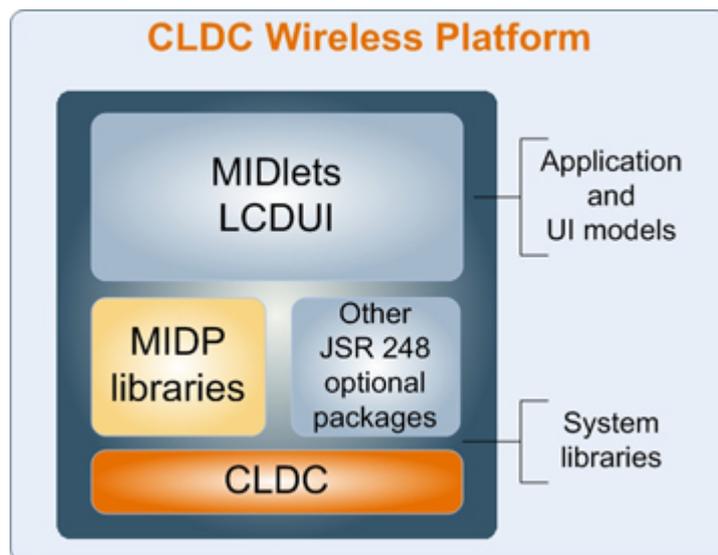


Figure 2: CLDC Wireless Platform

1.3.5.2 Binary Runtime Environment for Wireless (BREW)

BREW [38], invented by Qualcomm, provides an environment for mobile devices. It was first designed for use with CDMA devices, but has since been ported to other handsets, including GSM (Global System for Mobile communication) and GPRS (General Packet Radio Service). BREW is an environment for downloading and running small programs, such as playing games, sending messages, and sharing photos. The major benefit of the BREW environment is that the developers can easily port their applications between all Qualcomm devices. BREW-enabled handsets can be developed in C or C++ .

The developer community for BREW is, however, quite small. It is also not referred to in many books. Users should write their own solution for compressing resources with BREW. Compared to J2ME, commercial profilers for C/C++ are expensive.

1.3.5.3 Windows Electric Compact (Windows CE)

Windows CE [39] is a variation of Microsoft's Windows operating system for mobile handsets. Windows CE is designed for devices with minimal storage. The development tool [39], Visual Studio, supports projects for Windows CE /Windows Mobile to generate executable programs and platform images as an emulator, or attached by cable to a mobile device.

Most applications for mobile phone remote management utilise J2ME (Java 2 Platform Micro Edition) as a tool for user interface implementation. Yang and Kou [25] presented two techniques that use J2ME to monitor and control PC clusters from mobile phones.

Not only has J2ME been deployed in mobile applications, but also in other tools, such as Windows and C#. Nichols and Myers [4] set out to generate a smart phone interface generator using Microsoft's Windows CE-based Smartphone platform. These interfaces allow users to manage each appliance's full functionality and are consistent with other interfaces of the phone.

1.4 Embedded Databases

Databases have provided efficient information retrieval engines for a number of applications for decades. There are many mature databases for mainframes, servers, and even PCs, such as Oracle, MySQL, DB2, and Sybase. Many of them have been applied to applications.

There are, however, few applications that have embedded the database into a cell phone, due to the limitation of resources within the mobile phone. These limitations include the power source, network connection, and memory size.

In recent years, with the development of technology, the price of hardware has dropped significantly. The mobile phones have become more functional and powerful, and the embedded database can now be realised in handsets. Currently, there are a few embedded databases available for handsets, which include Perst Lite [40], PointBase [41], db4o [42], and RMS (Record Management System).

1.4.1 Perst Lite

Perst [43] is a simple, object-oriented, embedded database. It is easy to use and provides high performance compared with other databases for mobile phones. It is intended to be used in applications which need to deal with persistent data in a more sophisticated way than the load/store object tree provided by standard serialisation mechanisms. Perst also provides fault tolerant support; ACID (Atomicity, Consistency, Isolation, and Durability) transactions; and concurrent access to the database. Tight integration with programming language is the main benefit of Perst. Perst stores objects directly without packing/unpacking code (which has to be written for traditional relational databases), so there is no gap between the database and application data models. Also, Perst, unlike many other OODBMS (Object oriented database management system), does not require a special compiler or pre-processor and provides a high level of transparency. Perst Lite is becoming a particular embedded database for mobile phones.

1.4.2 RMS

RMS (Record Management System) is an API (Application Programming Interface) for storage data provided by J2ME MIDP. It can store, retrieve, and delete records such as a file system. It is a small database of simple, oriented-records [44].

1.4.3 Pointbase

PointBase Micro [41] is a platform-independent Java relational database optimised to run on the Java 2 Micro Edition (J2ME CDC (Connected Device Configuration) and CLDC/MIDP)) and J2SE platforms. It has an ultra-compact footprint (footprint size <45KB for J2ME MIDP) and can be easily embedded within a Java application, making it transparent to users from the time of deployment. PointBase Micro is designed for notebooks, PDAs, and emerging Java-enabled devices [41]. It provides effective data management for rapid and efficient mobile enterprise applications created by software vendors and systems integrators.

1.4.4 db4o

db4o [42] is an open source object database that enables Java and .NET developers to reduce development time and costs and enhance performance. The unique design of db4o's native object database engine is suitable to be embedded in equipment and devices, and in packaged software running on desktop platforms, mobile, and in real-time control systems - in brief: In all Java and .NET environments without a database administrator (DBA) [42].

Table 1 provides a comparison of Perst Lite, RMS, PointBase and db4o based on different features. Based on this comparison, Perst Lite is chosen as the database embedded into handsets. The Perst Lite J2ME database has simplicity in design and high performance within the resource limits of most intelligent mobile and embedded devices. Perst Lite contains B-tree, Patricia Trie, Bit index, T-Tree, and R-Tree indexes, as well as List, Relation, and Set collections, all protected by transactions supporting the ACID properties (Atomicity, Consistency, Isolation, and Durability). Perst Lite also offers additional features of multithreaded access, data encryption, and asynchronous replication. The query language of Perst Lite is not, however, a standard query language. In addition, for commercial purposes, Perst Lite costs US\$2000 per application each year.

Table 1: Database Comparison

	Pointbase Micro	Perst Lite	db4o	RMS
Size	45K for J2ME MIDP, 90K for J2ME CDC	30K~300K	250K	Default component of cell phone
Support language	JAVA	JAVA & .net	JAVA & .net	JAVA
Platforms	J2EE, J2SE, MIDP, personal JAVA	J2ME, J2SE	JDK 1.1 or later	All J2ME
Application platforms	PDA, mobile handset, PC	PDA, mobile handset	PDA, PC	PDA, mobile handset, PC
Database engine	Relational	Object-oriented	object-oriented database	None
Query language	Subset of SQL92	QBE, perst search method	QBE OR SOND	None
JDBC	JDBC subset	none	none	none
Utilities	Console, others	none	Defragment (not GUI)	none
Reflection	yes	none	no	none
Ease of use	good	Very good	Very good	none
Flexibility	Cross-platform	Cross-platform	Cross-platform	none
Performance	good	good	poor	Very poor
Open source	none	yes	yes	yes
Security	encrypt	encrypt	none	no
Strength	Super-small footprint	Easy installation	Easy installation	simple
Weakness	SQL and JDBC subset	Non-standard query mechanism	Non-standard query mechanism	Without index, slow search engine
Cost	US\$299	US\$2000 for per application per year	US\$100 for personal, US\$ 1000 for corporate	none

1.5 Mobile User Interface

The subject of how humans interact with computer systems is referred to as HCI (human-computer interaction). There are many disciplines that contribute to HCI, such as computing science, psychology, ergonomics, engineering, and graphic design [45].

The design of a user interface (also called a human-machine interface) is a key part of computer systems. The goal of a user interface is to ease the exchange of information between the user and the device (computer or system) to be controlled. A well-organised interface not only makes a work situation easier to understand, but also

reduces errors and, therefore, limits the scope of possible damage caused by interactions [45].

1.5.1 Human-machine Communication

As Olsson and Piani [46] pointed out, ergonomics is a comprehensive discipline, which studies how human capabilities are best utilised in workplace surroundings and how these surroundings can be configured to best adjust to human requirements. It consists of different fields, such as engineering, physics, physiology, and psychology. In control system applications, some features of ergonomics are essential. The user must understand how to approach a system, what to search for, what to expect, and must also be familiar with the general command principles for a new machine. The designer must define how data is illustrated on terminals and control panels and must define the features of commands provided by the user. So, the problem of the user interface from the perspective of user, or designer, is important to the control system engineer [46].

For a user interface design, Olsson and Piani [46] also stated that there are no predefined, handbook-written rules. Nevertheless, some principles can be followed to avoid major mistakes. Experience shows, however, that principles and trends, ergonomics, and computer user psychology are also fields for research. The features of the interface need to be correctly defined beforehand. It is estimated that 50 to 70% of the whole control system software is related to the generation of the user interface [46].

1.5.2 Psychological Considerations

According to estimates, one billion bits of information enter the human body per second, as described by Piani [47]. Despite this, only 100 bit/sec is processed consciously. The brain is likely to further reduce the amount of information to be processed. At the same time, if too much information is presented, human beings lose their capacity for action and their attention tends to concentrate on only one piece of the input data [47].

A model of human cognition and information processing is needed to define successful methods for the user interface design, as discussed by Piani [47]. Sensory storage, short term memory, and long term memory are the functional blocks identified by modern psychology in the cognitive process. Perception is the stage in the information processing conducted by the brain, including storage into the short and long term

memory, planning, and transferring of information into control action. The information from sense storage is transmitted to short term memory and, if there is a voluntary effort, will be transferred to the long term memory.

Miller [48] declared that 7 ± 2 information items, *chunks*, could be held in short term memory. The existing chunks will be erased, or displaced, by new incoming information. The new information will be handled more easily through organisation and the relationship with previous knowledge. The relationship to previous experience is generated using symbols from everyday life in a user-centred system [46]. Visibility has a direct, realistic application in a user interface. Simple models, possibly connected to general, daily experience should be chosen. Consistency means that a command will keep the equivalent meaning, which can help to transmit existing knowledge to new contexts. Feedback is another key feature of user-oriented design. When a command has been provided, there should be some reaction to identify the new action. Feedback is not only a confirmation, but also allows the user to understand if an action has to be halted, or reversed [46].

Olsson and Piana [46] also stated that perception facility, coding, organising, and structuring are three ergonomic features influencing the acquisition of information by the user. Perception facility depends on the hardware equipment; for example, a terminal screen key's features include brightness, reflection absence, colour contrast, and symbol size. Coding is a method to transfer information using symbols and clues. Organising and structuring illustrate information to relieve the user from the voluntary effort of creating a mental structure [46].

A good user interface draws the user's attention to key facts and allows prompt and accurate reactions on the basis of the information provided. Thus, if a user has to react rapidly to new information, the presented information should be logically organised and should not exceed five distinct items at the same level [49].

1.5.3 Hardware for User Interface Communication

There are several kinds of devices for user interface and the most important input/output (I/O) devices can be divided into four types; direct I/O devices, direct control pointing devices, indirect control pointing devices, and other devices [46]. The direct I/O devices cover screen terminal, keyboard, special function keys, printing terminal, printer, and

control panel. The direct control pointing devices include lightpen and touchscreen, while the indirect control pointing devices consist of mouse, trackball, and joystick. Finally, the other devices contain speech recognition systems, speech generators, and optical, acoustic alarms [46]. Although there are many choices for user interface hardware, the limitations of mobile phones are obvious. For example, only 12 keys plus a few functional keys are provided on mobile phones. Lightpen, touchscreen, and speech recognition systems are not available in all mobile phones. Thus, it is a challenge to use the small mobile phone screen for providing the user interface.

1.5.4 Interface Design Principles

Designing a user interface covers many fields and areas of knowledge.

1.5.4.1 General Principles

Olsson and Piani [46] stated that screen displays must be simple and without unnecessary information. The information content also needs to focus on the user. The important issue in presenting large amounts of complex data is accurate structuring and selection. Thus, the original data needs to be divided into smaller sets for presentation. They also pointed out that only one main concept should be shown on every screen in the most easily understood way possible [46].

Generally speaking, a good user interface should have the following characteristics; it should be adequate for the purpose, self-explanatory, and consistent at many levels [45]. In the interface, as stated by Stone, Jarrett, Woodroffe and Minocha [45], the key issue is that the user can get an immediate *feeling* that a command has been received and accepted, even though processing has not yet reacted to it.

1.5.4.2 Parameter Description Syntax

Olsson and Piani [46] mentioned that the information presented on the terminal screen can be divided into a lead text (fixed text) and a variable for the dynamic variable (actual text). The lead text should generate an expectation satisfied with the combination with the dynamic information. Also, the lead texts do not need to give complete information in a grammatical sense, whereas the combination with dynamic information should make complete sense [46].

The difference between static and dynamic information is that the lead texts are written with normal intensity, with the dynamic variables presented in high intensity on-screen displays [47]. In static texts the use of words with negative connotations should be avoided.

1.5.4.3 Screen Layouts

Piani [47] pointed out the principles for screen layout design, which include the following:

- Because eye movement is from left to right and a drawing developed along the horizontal direction is easier to notice than one drawn vertically, the process evolution must be represented from left to right;
- The layouts of the screen must be consistent in their manifestation. A layout can be divided into four sections, including work (application related), control section, message section, and static information areas;
- Graphing should be structured as proximity, symmetry, similarity, and grouping;
- Colours are an influential method to present information. Seven different colours are the maximum for a screen;
- One colour = one meaning, meaning that colours should be used consistently;
- Colours can also be used for the demonstration of functional states. For example, green represents an indication of security, permission, or correctness. Red means a state of alarm, danger, and prohibition. Yellow is related to a warning and the presence of some minor problem;
- Colour mixture must be agreeable and not fatiguing. Ease of perception differs greatly with different colour combinations;
- Make sure not to rely only on colours as a means of presenting vital information, because a large number of people are blind to some colours and cannot recognise them. Use redundancy to present the information;

- Drawing is the most natural representation for objects;
- Blinking is another method to call for attention, however, a text should never blink; and
- The display layouts must be interesting and motivate the users of the control system, rather than bore them.

1.5.4.4 Commands

Furthermore, Piani [47] stated that pushing buttons on a panel, or typing in command sequences on a keyboard, can implement the communication from user to machine. He described the following considerations regarding the design of a user interface:

- A command should describe a reference value for a condition which will later become equal to the actual value only if the actuators, control system, sensors, and physical process are correct;
- Actions following the same command should lead to similar results;
- A user interface should be clearly indicated for several input alternatives. Input commands and data should be checked instantly;
- The current, prior, most frequent, or safest, command should be illustrated as the default selection for a command;
- String commands which are typed on a keyboard should be as short as possible, without losing their meaning;
- Only a few mixtures usually make sense in an area where a text input is requested;
- A command typed from a keyboard needs some consideration and may lead to errors. So, before execution of a sensitive command, it is necessary to ask for confirmation;
- Highly sensitive and potentially unsafe commands should be checked with a PIN (personal identification number); and

- In case of an emergency, the application can be stopped immediately.

1.5.4.5 Menus

Piani [47] also mentioned that the menu design should consider the following principles:

- The menu structure should be obvious to the user;
- An 'Exit' command should be illustrated;
- The same abstraction level should be kept for the items in a menu;
- The number of choices should be limited, remember the 7 ± 2 rule [48];
- Take the unrelated questions out of the screen; and
- Use the same keys for similar functions on different menus.

1.6 Project Objective

This project investigates embedded databases and graphical interfaces for the remotely controlled mobile application – MicroBaseJ for MCS Ltd. The project aims at the development of an integrated database and GUI (Graphical User Interface) for typical 3G, or 2.5G, mobile phone with Java MIDP2 capability. This includes methods of data acquisition, mobile data and information communication, data management, graphical user interface for data presentation and remote user control. Support of phone delivered informatics will require integrated server and networking infrastructure research and development, to support effective and timely delivery of data for incorporation in mobile device-based informatics applications. A key research and development (R&D) challenge is to support effective and timely delivery of data for incorporation in mobile device-based informatics applications. Another important point in the project is how to develop an efficient graphic user interface on the small mobile screen.

This research investigates and analyses the architecture of a mobile monitoring system. The project aims at generic solutions that can be implemented in a number of commercial sectors, such as horticulture, building management and pollution/water management. The developed model is tested with data relevant to the horticulture area

of application. The system also addresses the main issues related to mobile monitoring, including realtime response, data integrity, solution cost, graphical presentation, and persistent storage capabilities of modern mobile devices.

1.7 Project Plan

This project (MicroBaseJ) is designed according to software engineering practices, and is managed using the agile method. To implement MicroBaseJ, there are four steps that need to be considered in software engineering, including system requirement analysis, system design, system testing, and evolution. To manage the software process, a time schedule needs to be organised before implementation.

Table 2 - The Schedule for MicroBaseJ

The stages for MicroBaseJ	Duration	Start Date	End Date
Stage 1. Research	35 days	19/02/2007	6/04/2007
1.1 Mobile application research	20 days	19/02/2007	16/03/2007
1.2 Embedded database research	15 days	26/02/2007	16/03/2007
1.3 User interface research	15 days	19/03/2007	6/04/2007
1.4 Communication protocols for mobile application	25 days	2/03/2007	5/04/2007
Stage 2. Requirement analysis	30 days	10/04/2007	21/05/2007
2.1 Requirement collection	10 days	10/04/2007	23/04/2007
2.2 High level requirement collection	10 days	24/04/2007	7/05/2007
2.3 Define software requirement	10 days	8/05/2007	21/05/2007
Stage 3. The generic solution for project	10 days	15/05/2007	28/05/2007
Stage 4. System design	39 days	29/05/2007	20/07/2007
4.1 Architecture design	5 days	29/05/2007	4/06/2007
4.2 Class design	8 days	5/06/2007	14/06/2007
4.3 States design	8 days	15/06/2007	26/06/2007
4.4 Database design	8 days	27/06/2007	6/07/2007
4.5 User interface & communication design	10 days	9/07/2007	20/07/2007
Stage 5. System implementing & Testing	111 days	6/07/2007	7/12/2007
5.1 Structure the order of the implementing system	10 days	6/07/2007	19/07/2007
5.2 Coding	60 days	20/07/2007	11/10/2007
5.3 Testing separate units	50 days	27/09/2007	5/12/2007
5.4 Combining test	10 days	26/11/2007	7/12/2007
Stage 6. Documentation	30 days	17/12/2007	25/01/2008
Stage 7. Thesis writing	90 days	28/01/2008	30/05/2008

Table 2 shows the schedule outline for MicroBaseJ. It describes all activities throughout all MicroBaseJ development. This project is not only a software engineering project, but also a research project. So, the first stage of this project is research. It includes research into mobile applications, embedded database, user interface, and their communication protocols. The estimate for this part of the project was 35 days.

The next stage is system requirement specification, which was estimated to span 30 days. Requirement collection, high level requirements and definition of software requirements are the major tasks in this stage. The third stage provides the generic solution for MicroBaseJ, and the following stage is the key element in this project, system design. It contains architecture, class, state, database and user interface, and communication protocol design. It took 39 days to complete.

System implementation and testing follows system design, and lasted about 111 days. This section includes coding not only for MicroBaseJ, but also for embedded database testing and performance comparison. This section covers the implementation, testing, and evaluation.

As mentioned above, the agile method is intended to be used for this project management. It covers requirement analysis, system modelling, and testing. Agile software development is an incremental and iterated software development [2].

2 Development Environment and Tools

2.1 Introduction

This chapter describes the environments and tools for the project. The environment section introduces the hardware for development, simulation, and testing, including the mobile phone, computers, and the application platform for this project. The chapter also discusses the software tools for this application, including the IDE (Integrated development environment), J2ME, and a third party software, J2ME Polish.

2.2 Environment

2.2.1 Development Environment

The development environment of MicroBaseJ included two computers one desktop and one laptop as well as one mobile phone (a Nokia N73). Figure 3 shows the development environment of this project.

The development desktop was Dell with Pentium D CPU 1400MHz, 256 MB of RAM, with the operating system being Microsoft Windows XP Professional Version 2002. Another computer for this application was a laptop, Dell Latitude D600, with 1400MHz, 512MB RAM, and an operating system of Microsoft Windows XP Professional Version 2002.

These two computers makeup the platform used for the development, testing, and evaluation. They both run software such as JAVA JDK (Java Development Kit) (jdk-6u2-windows-i586-p), Eclipse (eclipse-sdk-3.2.2-win32), EclipseME (eclipseme.feature-1.7.8), J2ME Toolkit (sun_jave_wireless_toolkit-2_5_1), J2ME Polish (de.enough.mepose.ui_0.7.1), and Perst Lite (perst269). Java, J2ME, Eclipse, and J2ME Polish will be discussed in a later section of this chapter. Perst Lite was investigated in Section 1.4.1 .

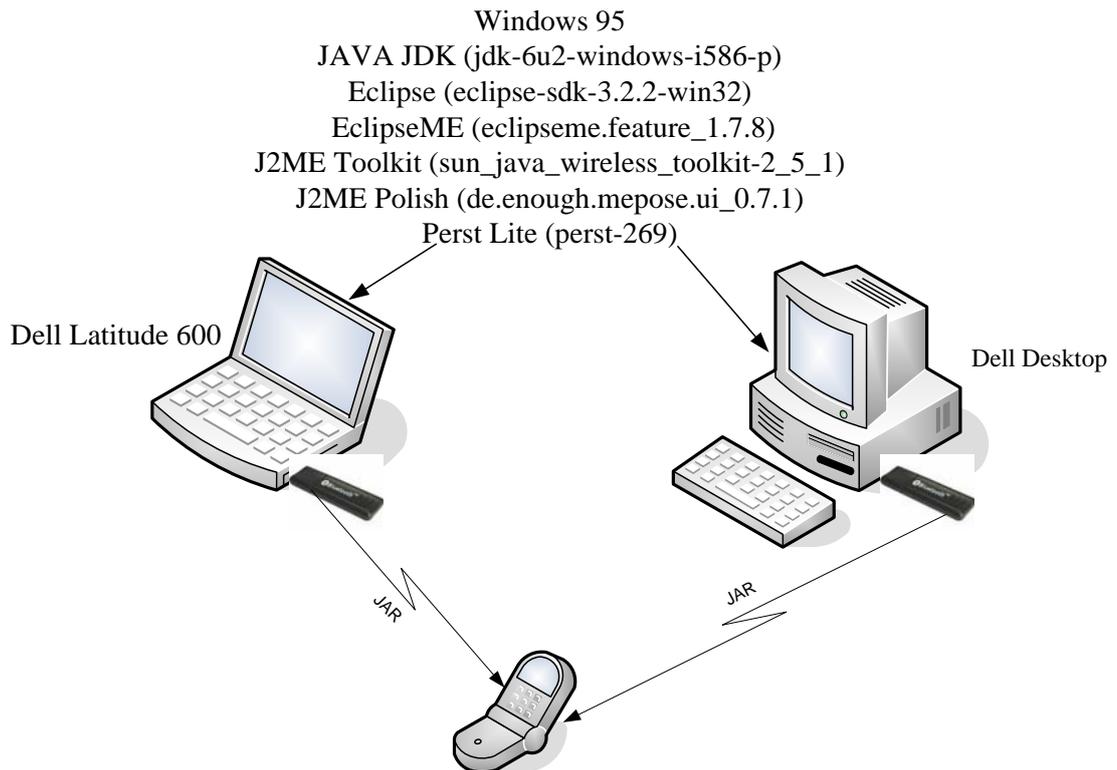


Figure 3: The Development Environment for MicroBaseJ

The application was developed and tested on both Dell computers. Once the code was found to work, the application's JAR (Java archive) file was uploaded to the mobile phone (Nokia N73) using Bluetooth protocol through a Bluetooth dongle.

2.2.2 Mobile handset

A Nokia N73 was chosen as the mobile handset to test the performance of the embedded database and the project.

Nokia N73 adopts Symbian version 9.1 as the operating system and S60 3rd Edition as the user interface. It supports WCDMA (Wideband Code Division Multiple Access) 2100/EGSM (Extension of GSM) 850/900/1800/1900 networks. This N73 has 42 MB internal dynamic memory plus 1G miniSD (Secure Digital) memory card (hot swappable) for contacts, text messages, multimedia messages, ringing tones, images, video clips, calendar notes, to-do list, and applications. It also has video, photos, and music functions. The N73 screen is a large, bright 2.4 inch QVGA (Quarter Video

Graphics Array, 240 x 320 pixels) TFT (Thin-film transistor) colour display with up to 262,144 colours.

The Nokia N73 provides GPRS (General Packet Radio Service), EDGE (Enhanced Data rates for GSM Evolution), and UMTS (Universal Mobile Telecommunications System) interfaces. It also provides Internet browsers, XHTML (eXtensible HyperText Markup Language) and HTML (HyperText Markup Language).

The Nokia N73 supports MIDP 2.0 and CLDC 1.1, 3D (three dimensional) API, PIM (Personal Information Management) API, File access API, C++ and Java SDKs (Software development kit). MicroBaseJ was developed as a J2ME application deployed on Nokia N73 to evaluate the embedded database performance and the project performance.

2.2.3 Application Platform

Figure 4 illustrates the application platform for the project. The MicroBaseJ application contains three major components; the Cellular Sentinel, the mobile phone, and the server. The Sentinel is used for information collection from the sensors, the information includes temperature, humidity, soil moisture, wind direction and speed, rainfall, and leaf wetness. These data are also sent to the mobile phone by the Sentinel. The mobile phone receives data from the Cellular Sentinel and stores these records into the embedded database. A user can retrieve and present these records as graphs on the small phone screen. When the record number exceeds a particular limit, the earlier incoming records can be sent to the server via Bluetooth and be deleted from the embedded database. The server receives the backup data from the mobile phone and saves them as a backup file. The following section introduces the Cellular Sentinel. The mobile phone can be any cell phone with MIDP 2.0 capability. The server can be any computer with a large storage capacity.

The Cellular Sentinel is a pocket based informatics for asset management and remote control solution of preference. The Cellular Sentinel uses TINI DS80C400 and Wavecom Fastrack hardware.

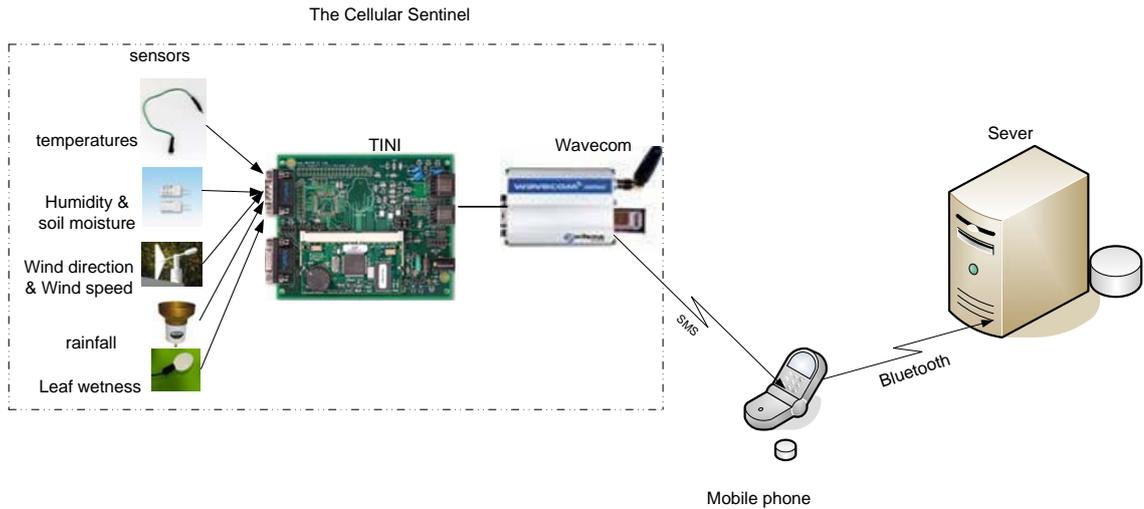


Figure 4: The MicroBaseJ Platform

2.2.3.1 TINI

The TINI reference board is based on the DS80C400 processor [50]. The DS80C400 includes 1M RAM, 1M flash, 144-Pin SODIMM (small outline dual in-line memory module), and the networking includes 1-Wire, serial, and 10/100 connections.

The TINI [50] provides physical connectors to interface with other devices such as Ethernet networks, serial devices, and a 1-Wire network.

The developers can choose difference languages, such as Java, C, or even coding in 8051. It is a friendly, remote asset management and control system [50].

2.2.3.2 Wavecom Fastrack

The Wavecom Fastrack modem provides immediate wireless capabilities. Housed in a rugged metallic casing, the Fastrack modem is built to endure the toughest environments. Its open interfaces, and OpenAT commands can be embedded into it, with the applications being able to be run right on the WISMO (Wireless Standard Module) platform. It is also GSM/GPRS enabled [51].

The Sentinel can collect information-monitoring electronics and sensors. The application of the Cellular Sentinel can be suitable for security, irrigation,

pollution, GIS, temperature, weather, crime, telemedicine, and remote vision applications. It is compatible with GIS, Bluetooth, and mesh networks.

The Sentinel uses 1-Wire to communicate with the sensors and to collect the relevant sensor information, with this information edited with a time stamp (adding year, month, day, hour, minute, and second), then being ready to be sent to the mobile phone by the Wavecom Fastrack.

2.3 Tools

2.3.1 IDE environment

An IDE is a software application providing wide-ranging facilities to develop software for computer programmers. An IDE usually contains an editor for source code, a compiler (and interpreter), tools for build automation, and a debugger. An IDE may include a version control system and different tools to simplify the GUI construction [52].

There are a number of IDEs available, such as Eclipse and NetBean. Eclipse is chosen as the development environment for this project due to its better performance and features; for example, it has faster initialisation compared with NetBean.

Eclipse is an open source community for construction of Java-based tools and structures to help programmers solve their problems [53]. The schemes of Eclipse are focused on creating an extensible development platform, runtimes, and application frameworks for building, deploying, and managing software across the entire software lifecycle, including enterprise development, embedded and device development, rich client platforms, rich Internet applications, application frameworks, application lifecycle management, and service oriented architecture [52]. The Eclipse version of this project is Eclipse DSK (Development Software Kit) 3.2.2.

2.3.2 Software

It is very important for a programmer to choose an appropriate language for software application development.

Java, which has been developed by Sun Microsystems, contains various software products and specifications that provide a system for application software development and deployment in a cross-platform environment [54]. Java can be used in a number of computing platforms, including low-end devices to enterprise servers (such as embedded devices and mobile phones), and even high-end devices (such as supercomputers). Java can be considered as ubiquitous in mobile phones, Web servers and enterprise applications, and desktop computers [54].

Java syntax is quite similar to C and C++, but it gets rid of certain low-level constructs such as pointers and has a simple memory model to allocate every object in the heap and refer to all variables of object types. JVM (JAVA virtual machine) deals with memory management through integrated automatic garbage collection.

Java has the following benefits: It starts quickly; it writes less code; it writes better code; it develops programs more quickly; it avoids platform dependencies; it writes once and runs anywhere; and it distributes software more easily [55].

There are three Java platforms defined by Sun Microsystems; J2SE, J2EE, and J2ME [54]. J2SE can be used for developing and deploying Java applications on desktops and servers. J2SE contains two major parts; Java SE Runtime Environment (JRE) and Java SE Develop Kit; of the Java SE platform family [35]. J2EE, built on the foundation of J2SE, is the industry standard for implementing enterprise-class service-oriented architecture (SOA) and next-generation web applications. J2ME aims to provide a group of Java APIs for the development of resource-constrained devices, including mobile phones and PDAs [56]. The Java platform in this project is jre1.6.0_01.

2.3.2.1 J2ME

Java ME technology can deal with the constraints associated with building applications for small mobile devices. Java ME technology allows programmers to generate Java applications running on small devices with limited memory, display, and power capacity based on a limited environment. Java ME is a set of specifications and technologies that can be used for the construction of a complete Java runtime environment for the requirements of a small mobile device [37].

The Sun Java Wireless Toolkit is a set of tools for generating Java applications that run on devices obedient with the Java Technology for the Wireless Industry (JTWI, JSR 185) specification and the Mobile Service Architecture (MSA, JSR 248) specification. It includes build tools, utilities, and a device emulator [37].

This project adopts the Sun Java Wireless Toolkit, WTK2.5.1. WTK2.5.1 contains all of the advanced development features found in WTK version 2.2, 2.3 Beta, and 2.5 Beta 2. For example, it contains MIDlet signing, certificate management, integrated over-the-air (OTA) emulation, and push registry emulation.

J2ME provides a graphic user interface API for programmer; including low level, Canvas, and high level API, Screen. Screen contains Form, List, Alert, and TextBox subclasses. The latter three items are the predefined components, whereas the former is the open type. It is a container to support multiple items. The programmer uses the low level API; that is, Canvas's paint method; to draw a screen picture. High level API can help the programmer to develop user interfaces to be faster and more portable [57].

2.3.2.2 J2ME Polish

The programmer cannot change the presentation of the GUI defined by high level API and low level API. For example, it does not allow the user to define sharpness, colour, and font. Although the programmer can also use low level GUI API to generate a good-looking interface, it needs extra work to allow the user interface to run on all, or most, J2ME devices. Thus, the third party software J2ME Polish is proposed.

The free third party software, J2ME Polish, which was developed by Virkus, is a set of open source tools for generating *polished* wireless Java applications [58]. J2ME Polish provides its build tools and user interface for the programmer. The J2ME Polish GUI is not only compatible with the high level GUI API, but also allows the programmer to design the detail of the user interfaces. The J2ME Polish has the following unique characteristics; easy implementation, automatic porting, innovative designs, customisation, flexibility, and extensibility. J2ME Polish provides several features; such as circumventing device bugs and integrating the

best matching resources; to help the programmer solve difficulties in the real world of wireless Java application [58].

J2ME Polish tools can be divided into four layers; build framework, client framework, IDE plug-ins, and stand-alone tools. The build framework includes pre-processing, compiling, pre-verification, packaging, device database, logging, and localisation. Programmers can use it to build their J2ME applications. The client framework contains high-level GUI, game engine, logging framework, WMAPI (wireless messaging API), and utility classes. Programmers use APIs from the client framework to enhance their wireless Java applications. The IDE plug-ins allows programmers to develop their J2ME application in the popular Eclipse IDE. It consists of Eclipse, NetBeans, a preprocessing editor, debugging, emulator invocation, and quick device selection. Finally, there are some stand-alone tools, such as a binary editor, a bitmap-font editor, and a project manager [58].

2.4 Summary

This chapter discusses the hardware environments and the software tools used to develop/implement MicroBaseJ. The application platform is based on three components: the Cellular Sentinel, the mobile phone, and the server. The Cellular Sentinel includes sensors, TINI, and Wavecom Fastrack. The development environment is based on two Dell computers and Nokia N73 mobile phone. This chapter also introduces and discusses the software tools used; including Eclipse, J2ME, and the third party software, J2M2 Polish.

3 System Design

3.1 Introduction

This chapter investigates the system design for MicroBaseJ in terms of software engineering. It analyses the requirements of the project and results in the general architecture. Next, the data sources and communication protocols are described. In addition, the functional and non-functional requirements are provided. Interface descriptions, data structures between components, and the dataflow model of the project are also discussed. Furthermore, class diagrams and state diagrams are proposed and discussed in terms of UML.

3.2 Requirement Analysis

The specification is one of main components in software engineering. Sommerville [59] introduced the user requirement and the system requirement into the requirement analysis. The user requirement can be considered as being a high level abstract requirement from the users. The system requirement provides the detail of the system services and its constraints. The requirement process needs to be dealt with through the system designer and customers.

3.2.1 User Requirement

From the viewpoint of MCS Ltd, this project relates to the MCS Mobile application framework for producing quality, portable, compatible, and appealing applications.

MCS aims to develop a generic solution for the mobile remote management. The first step focuses, however, on the horticultural field, and is designed to explore an efficient embedded database of mobile phones and develop applications based on this prototype. The embedded database will store the data gathered from remote sensors, including temperatures, humidity, soil moisture, wind speed and direction, rainfall, and wetness. Furthermore, this application also aims to develop a friendly and well-organised user

interface to present data collected from sensors. The users of the application include MCS staff/contractors, farmers, greenhouses, and cool room managers.

3.2.2 Data Sources

The needed data of this project includes four temperatures (three environmental temperatures and one soil temperature), wind speed, wind direction, humidity, soil moisture, rainfall, and leaf wetness.

Temperature [60] is a physical property of a system that underlies the common notions of hot and cold; something that is hotter generally has the greater temperature. Specifically, temperature is a measure of the kinetic energy of a sample of matter. Temperature is one of the principal parameters of thermodynamics. In this project, there are four temperature items. Three of them monitor the environment and plants. One is used for monitoring soil temperature. In MicroBaseJ, the data range of the temperatures collected from sensors is -55.00 to +125.00 Celsius, with a resolution of 0.01 Celsius.

Humidity is referred to as relative humidity. Relative humidity is described as “the (dimensionless) ratio of the actual vapor pressure of the air to the saturation vapor pressure” [61]. It is expressed as a percentage. Relative humidity is one of the important parameters in forecasting weather. MicroBaseJ collects relative humidity from the sensor that monitors the environment of grown plants. In MicroBaseJ, the humidity collected from the sensor varies from 0.00% to 100.00%, with resolution of 0.01%.

Soil moisture is quite similar to humidity. It is used for measuring the quantity of water contained in soil on a volumetric, or gravimetric basis [62]. Its unit is a percentage. Soil moisture in MicroBaseJ is used for monitoring the soil water content to protect grown plants. The soil moisture collected from the sensor varies from 0.00% to 100.00% in MicroBaseJ.

Wind speed [63] is defined as the movement of air from one place to the next. Wind speed is an important factor in weather forecasting. In MicroBaseJ, wind speed is chosen to work out where plants will be best protected from strong winds. The data range of wind speed collected from the sensor is from 0 to 200 Km/h.

Table 3: The relation of values and wind direction

values	Wind direction
1	north (N)
2	north north west (NNW)
3	north west (NW)
4	north west west (NWW)
5	west (W)
6	west west south (WWS)
7	west south (WS)
8	west south south (WSS)
9	south (S)
10	south south east (SSE)
11	south east (SE)
12	south east east (SEE)
13	east (E)
14	east east north (EEN)
15	east north (EN)
16	east north north (ENN)

Wind direction [64] is the measurement of the direction from which the wind is blowing. It is usually described in terms of the essential direction, or in degrees of azimuth. Wind direction can be used for identifying where to best protect plants from strong winds. The data range of wind direction from the sensor in MicroBaseJ varies from 0 to 15. Digit 1 represents north (N); digit 2 represents north north west (NNW); and so on. Table 3 describes the relation of the values and the wind directions.

Rainfall [65] is a measurement that calculates the amount of rain falling to the ground. It is described as the depth of water collected on a smooth surface, and is normally calculated with accuracy up to 0.1 mm. The value range of the rainfall is measured between 0 to 200mm, with resolution of 0.1mm.

Leaf wetness [66] is used for the description of the plant's moisture in a canopy, because leaves are the essential organs of plant and are infected by most pathogens. In this project, leaf wetness is collected from a sensor which is placed on the leaf surface. The health of the plant can be estimated by combining leaf wetness with other parameters such as temperature and humidity. There are two values set in this project, digit 0 and 1. Digit 0 represents dry, and digit 1 is wet.

3.2.3 The Proposed Structure for MicroBaseJ

According to the user requirement mentioned above, a basic structure for the remote monitoring application has been proposed as shown in Figure 5. It consists of the Cellular Sentinel, a mobile phone, a server, and the public wireless network. The Cellular Sentinel collects the information from the sensors, and sends these data to a mobile phone. The mobile phone receives the notification information from the Cellular Sentinel, including temperature, humidity, soil moisture, wind speed and direction, rainfall, and leaf wetness. If any parameter exceeds the preset alarm limit, an alert message is generated on the mobile phone screen immediately. The relevant data from the Sentinel are also stored in the mobile phone's database locally. One key issue of the project is to investigate an embedded database for the mobile phone.

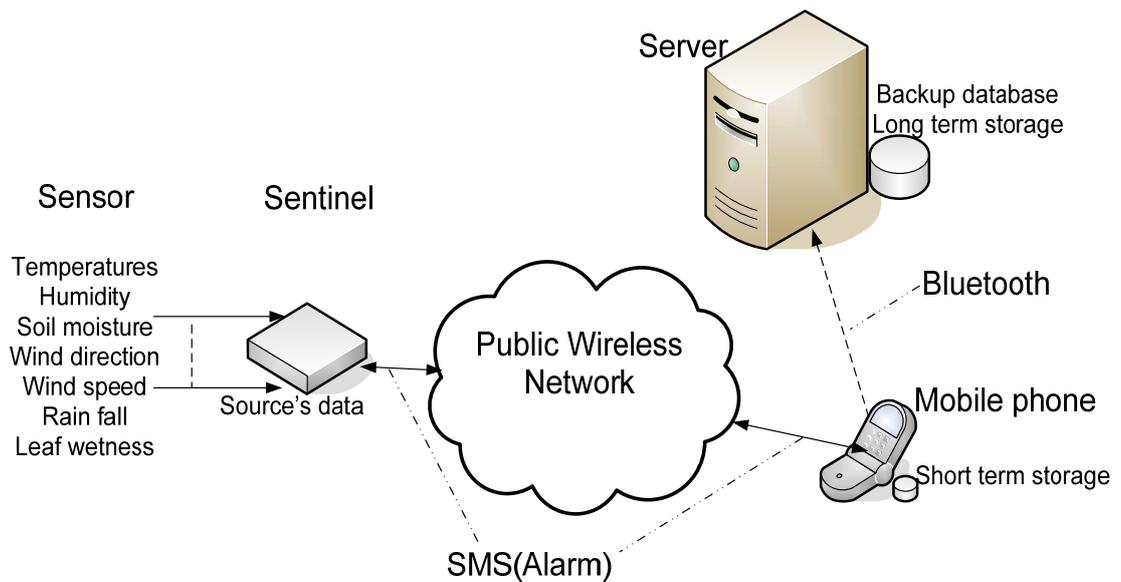


Figure 5: The architecture of MicroBaseJ

The mobile phone also provides a user interface for the subscriber to make enquiries to the local database and plots the relevant data locally. Users access the data from the database embedded into the mobile phone, through a friendly user interface. The data are presented as discrete readings, or as trend records, on the mobile screen. The user interface design presents another challenge in this application, due to resource-

constraints of the mobile phone. This application aims to discover a universal user interface for all MIDP2 mobile phones.

The relevant data about temperature, humidity, wind speed and direction, rainfall, and leaf wetness are sent to the server when the total amount in the database exceeds 18,000 records. This can prevent the mobile phone from crashing and improves the efficiency of the mobile phone. The server receives data from the mobile phone's embedded database and saves them in the server's database as a backup.

3.2.4 Communication Design

There are many choices for communication protocols in the project. However, finding an appropriate one is a challenge. There are a number of features to be considered, such as cost, flexibility, and maturation.

MicroBaseJ uses SMS as the communication protocol to transmit the collected data to the user handset, as it has many advantages. SMS has been widely utilised in the wireless world. All mobile phones provide SMS API and it is a cost-effective protocol. Vodafone in New Zealand provides customers with a special service (BestMate), whereby a mobile phone can text to the preselected handset an unlimited number of times, costing a total of only \$6 per month. Last but not least, the Sentinel has a GSM module installed in it which can send the collected data to the mobile phone.

On the other hand, the communication protocol chosen for backup of the data from the mobile phone to the server is Bluetooth. This protocol is chosen for several reasons. First, the backup is not a real-time operation. Next, it can be done as a one to one transfer over a short distance. Finally, the most important reason is that the communication method is free.

3.2.5 Functional Requirement

A system functional requirement provides the functionality, or services, expected for the system. To satisfy the user requirements of MicroBaseJ, 17 sub-functions are defined. The ID is labelled as MB (represent MicroBaseJ) plus a sequence number. Table 4 describes the 17 functional requirements, with IDs and comments.

The functional requirements include; Data Require & Send, Alarm Reset, Database Design, Data Stored, Alarm Present, Main Menu, Graphing Main Menu, Graphing Second Menu, Data Retrieved, Data Present, Sample Main Menu, Sample Confirm Menu, Alarm Main Menu, Alarm Reset Menu, Alarm Update Confirm Menu, Backup Data, and Backup Data Stored. Each function carries out a particular task. For example, the function that the mobile phone receives the normal collected data as background, and persists the information to the embedded database, is carried out by the Data Stored function. All the details of the functional requirements are shown in Table 4, below.

Table 4: The functional requirements for MicroBaseJ

ID	Requirement	Comment
MB.1	Data Require & Send – the Sentinel requests the data from the sensors. The Sentinel sends the notification collected from the sensors to the mobile phone.	OpenGL /SMS
MB.2	Alarm Reset – the Sentinel receives the reset sample rate from the mobile phone and updates the internal configuration accordingly.	OpenGL /SMS
MB.3	Database Design – designs the database structure for the information from the Sentinel	Perst Lite
MB.5	Data Stored – the cell phone receives the normal collected data as background and persists the information to the embedded database.	J2ME /Perst Lite
MB.6	Alarm Present – the J2ME application must be aware of incoming alarms and display an alert on the mobile phone screen.	J2ME
MB.8	Main Menu – once the user activates the application a friendly interface is shown on the mobile screen, allowing the user to manipulate the data charting and configuration-setting menu.	J2ME
MB.9	Graphing Main Menu – once the user chooses the charting option, a list of options is shown to allow the user to select two charting parameters.	J2ME
MB.11	Graphing Second Menu – once the user chooses two items, a form occurs to allow the user to type in the shown period.	J2ME
MB.12	Data Retrieved – the particular data are retrieved from the embedded database in terms of the user requirement.	J2ME / Perst Lite
MB.13	Data Present – the selected data are presented on the mobile screen (charting).	J2ME
MB.14	Sample Main Menu – once the user chooses the sample rate configuration option, the old sample rate is presented and allows user to type in a new one.	J2ME
MB.15	Sample Confirm Menu – asks the user to confirm the new sample rate and sends it to the Sentinel.	J2ME

MB.16	Alarm Main Menu – once the user chooses the alarm configuration option, a list of options for different items of alarm setting are provided for selection.	J2ME
MB.17	Alarm Reset Menu – provides a form to the user to update the alarm setting.	J2ME
MB.18	Alarm Update Confirm Menu – asks the user to confirm the change of the alarm data and stores it into the mobile phone.	J2ME / SMS
MB.19	Backup Data – provides a form for the user to confirm to backup the first in 5,000 records to the server and delete them.	J2ME
MB.20	Backup Data Stored – stores the backup data from the mobile phone into the database in the server.	J2EE / MySQL

3.2.6 Non-Functional Requirement

The non-functional requirement defines the requirements that are not directly related to system specific functions. Table 5 shows the non-functional requirements from MCS Ltd. MCS Ltd suggested that the application was run on mobile phones with MIDP 2.0 capability and expected to investigate an embedded database for MicroBaseJ. Before the embedded database can be applied to the application, the performance of the databases needs to be evaluated. Furthermore, a user-friendly interface needs to be considered either using third-party software, or customising. The final factor is the Sentinel, for which the TINI was chosen.

Table 5: The Non-Functional Requirements for MicroBaseJ

ID	Requirement	Comment
MB.21	MIDP 2.0	The use of MIDP 2.0 will allow MCS to deliver a more compelling user experience.
MB.22	J2ME architecture that will allow the possibility of other third party UI API's to be used in the future.	Attractive and appealing user interface.
MB.23	Embedded database – Perst Lite evaluation.	Need to evaluate the performance of Perst Lite and RMS and make a decision on the use of fast search techniques for the user. Also, analyse the application foot print.

3.2.7 MicroBaseJ Roadmap

To manage this project in terms of the agile method, the application is divided into four release versions, Release 0.1.0, Release 0.2.0, Release 1.0.0, and Release 1.1.0. Each release implements particular functions, like a milestone in the project. The programmer implements the first release, and tests it. It can be considered to be a relatively independent function of the project. When one release is finished, another is implemented. The advantage of the agile method is that the function can be implemented quickly and is adjustable. Users can check whether it has satisfied the requirements in time, and this helps the programmer to make a decision as to whether or not there is a need to further improve, or continue. This feature allows the software development to adapt the updated requirement in a short time frame.

A roadmap of MicroBaseJ is shown in Table 6. This table describes the deadlines, revision versions of the application, and their functions for each release.

Each release represents a milestone in the application. For example, Release 0.1.0 implements the first step of this project. It covers the embedded database discovery, analysis, and sensor data collecting, sending and receiving. Release 0.2.0 realises basic functions for MicroBaseJ, including the user interface and alarm notification. Configurations such as alarms and the sample rate are implemented in Release 1.0.0. The last one, Release 1.1.0, backs up the first-in data to avoid mobile phone crashes due to data overflow.

Table 6 shows that each release version contains several revisions. Each revision implements a specific function. For example, Revision 0.2.1 allows the user to update the alarm settings and sample rate settings, with the updated information stored in the local file system, or sent to the Sentinel for the next data collection.

Table 6: MicroBaseJ Roadmap

Release version	Due date	Revision version	Task	ID	Implement location	Contents
Release 0.1.0	Mid July, 2007	Revision 0.0.1	Data Storage Analysis	MB.23	Simulation	Embedded database – Perst Lite evaluation

		Revision 0.0.2	Mobile Device Performance Analysis	MB.23	Mobile phone	Perst Lite evaluation
		Revision 0.0.3	Data Delivery	MB.3	Mobile phone	Database Design
	MB.1			Sentinel	Data require & send	
	MB.5			Mobile phone	Data stored	
Release 0.2.0	Early September, 2007	Revision 0.1.1	User Interface Design for Mobile Phone	MB.8	Mobile phone	Main Menu
				MB.9	Mobile phone	Graphing main menu
				MB.10	Mobile phone	Graphing second menu
				MB.11	Mobile phone	Data retrieved
				MB.12	Mobile phone	Data present
		Revision 0.1.2	Alarm Notification Delivery	MB.6	Mobile phone	Alarm present
Release 1.0.0	Late October, 2007	Revision 0.2.1	Configuration Process	MB.14	Mobile phone	Sample main menu
				MB.15	Mobile phone	Sample Confirm menu
				MB.16	Mobile phone	Alarm main menu
				MB.17	Mobile phone	Alarm reset menu
				MB.18	Mobile phone	Alarm update confirm menu
		Revision 0.2.2	Configuration Delivery	MB.2	Sentinel	Alarm reset
Release 1.1.0	Late May 2008	Revision 0.3.1	Backup Delivery	MB.19	Mobile phone	Backup data
				MB.20	Sever	Backup data stored

3.2.8 Interface Description

The application contains three components; the Sentinel, the mobile phone, and the computer server. Figure 6 illustrates the communication among these three components. The Sentinel sends the notification collected from the sensors to the mobile user every hour, and receives the updated sample rate from the mobile user. The mobile user receives the data from the Sentinel and sends the new sample rate setting to the Sentinel. The mobile phone also sends the backup data; the first-in 5,000 records; to the server when the amount in the embedded database exceeds 18,000 records and deletes

them. The server receives the data from the mobile user and stores them into the database in the server.

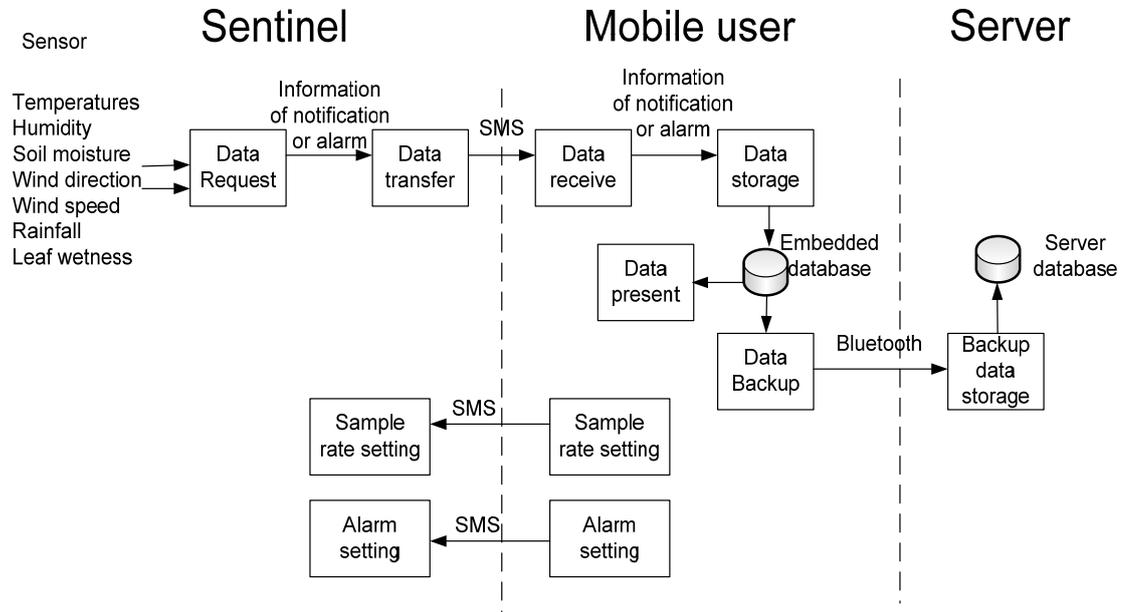


Figure 6: The Interface Diagram for MicroBaseJ

3.2.9 Data Structure between Components

To implement the functional requirements mentioned in Section 0, the application needs to define the data format between the Sentinel and the mobile phone, and the mobile phone and the server.

Table 7 defines the data format between the Sentinel and the mobile phone. This application supposes that the Sentinel sends data to the mobile phone each hour. The data includes the time stamp with four temperature readings, two humidity readings, wind speed and direction, rainfall, and leaf wetness. There are eleven attributes. All data are set as string and each attribute has a format of “title + ‘=’ +value (String type)”. For example, the temperature 1, t1, is 24.45 °C, and is set as T1=24.45. There is a semicolon among each attribute. An example of the whole data format is:

```
DATETIME=22020802000000;T1=23.73;T2=12.34;T3=23.55;ST=15.34;SM=67.45;R
F=23.24;HD=70.45;WD=12;WS=13.23;LW=1
```

where DATETIME is the time stamp for each record; T1, T2, T3, and T4 represent temperatures t1, t2, t3, and t4; SM is soil moisture; RF is rainfall; HD stands for humidity; WD and WS are wind direction and wind speed, respectively; and LW is leaf wetness. The detailed explanations can be seen in Table 7.

Table 7: Data Format for Messages from Sentinel to Mobile Phone

name	data type	Command format	range	Unit	Comment
date time	string	DATETIME= DDMMYYHHNNSSZZ (DD: day; MM: month; YY: year HH: hour NN: minute SS: second ZZ: ms)	DD: 1~31, MM:1~12, YY:00~99, HH:1~24, NN:0~59, SS: 0~59, ZZ: 0~59		ID for each sample data
space	string	","			
t1	string	T1=XXXX.XX	-55 ~ +125	°C	The first temperature
space	string	","			
t2	string	T2=XXXX.XX	-55 ~ +125	°C	The second temperature
space	string	","			
t3	string	T3=XXXX.XX	-55 ~ +125	°C	The third temperature
space	string	","			
st	string	ST=XXXX.XX	-50 ~ +50	°C	soil temperature
space	string	","			
sm	string	SM=XX.XX	0 ~ 100	%	soil moisture
space	string	","			
rf	string	RF=XXXXXX	0 ~ 9999	mm	rainfall
space	string	","			
hd	string	HD=XX.XX	0 ~ 100	%	humidity
space	string	","			
wd	string	WD=XX	1 ~ 16 (1:N, 2:NNW, 3:NW, 4:NWW, 5:W, 6:WWS, 7:WS, 8:WSS, 9:S, 10:SSE, 11:SE, 12:SEE, 13:E, 14:EEN, 15:EN, 16:ENN		wind direction
space	string	","			
ws	string	WS=XXX	0 ~ 200	km/h	wind speed
space	string	","			
lw	string	LW=X	w: wet; d:dry		leaf wetness

For the backup data from the embedded database to the server, the format is similar to the data from the Sentinel to the mobile phone, except for the DATETIME item (only 10 bytes in backup format) and the date type (the value in the backup is the integer type, whereas the value in the Sentinel is the float type).

Table 8 presents the format of the updated sample rate between the mobile phone and the Sentinel. The format uses the String type, set as SAMPLE=MMM, where MMM is the value of sampling. The unit is one minute. An example is: SAMPLE=30; meaning that the sample rate is 30 minutes.

Table 8: Sample Rate Format for Configuration to the Sentinel

name	data type	Command format	range	Unit
sample rate	string	SAMPLE=MMM	MMM:1~999	minute

3.2.10 Dataflow Model

According to Sommerville, the user requirement is usually described in natural language, as the target user may not be a technical expert [67]. A set of system models has been widely used for the documentation of the system specification; including the context model, the behavioural model, the data model, and the object model. The behavioural model defines all behaviours of the system. The dataflow model and the state machine model are two types of behavioural models. A dataflow model is used for the system driven by data, while a state machine model is suitable for a real time event driven system.

From the viewpoint of MicroBaseJ, the application is related to different components, such as the Sentinel, the mobile phone, and the computer server. Thus, a dataflow model is described in Figure 7. The figure also identifies sub-functions mentioned in Table 4 and Table 5. For example, the Data receive and alarm present process includes the sub-functions MB4, MB6, and MB7, as defined in Table 4.

Figure 7 shows that the Data request process collects the sensors' data and fills in the time stamp, and then the Sentinel sends it to the mobile phone according to the sample rate. The mobile phone receives data from the Sentinel and this process is run as background. If any parameter's value exceeds the alarm setting, an alarm form is shown on the screen and notifies the user with specific sound, meanwhile these data is saved into the embedded database. Otherwise, the process of saving the relevant data in embedded database is run as background. Otherwise, the process of saving the relevant data in embedded database is run as background.

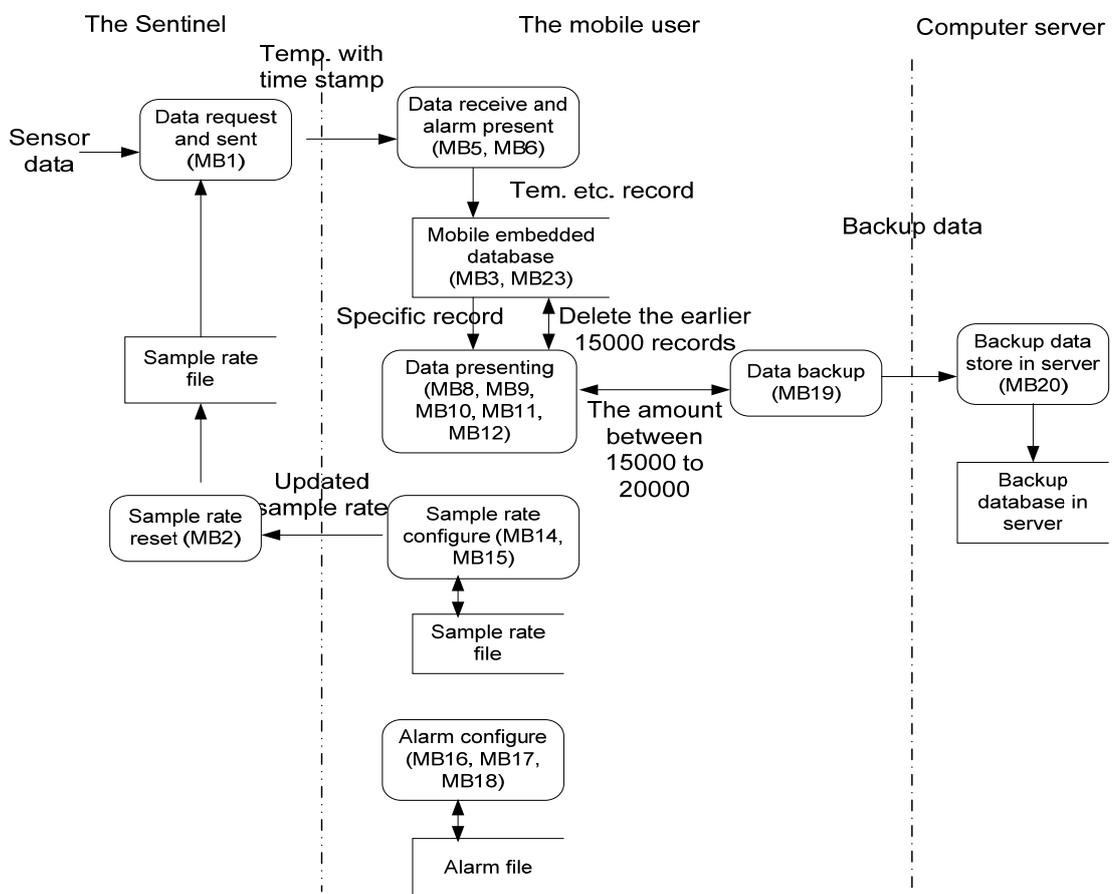


Figure 7: The Dataflow of MicroBaseJ

The mobile user can activate the application and select two types of attributes from the database according to the particular date chosen by the user. The relevant data is presented as a line chart, or bar chart, on the mobile screen. The mobile user can also configure the alarm, or the sample rate. The updated alarm, or sample rate, will be

stored in the respective files in the mobile phone and the modified sample setting is sent to the Sentinel. Furthermore, if the user activates the application, a confirm form is shown on the mobile screen to notify the user to set up the Bluetooth connection with the computer server when the number of records exceeds 18,000 records. The first-in 5,000 records is sent to the server and deleted from the embedded database if the user confirms the connection to be ready. Periodically deleting records can prevent the application from crashing as a result of the memory overloading.

The Sentinel receives the modified sample rate and saves it into the Sentinel. The computer server receives the backup data from the mobile phone and stores them into the database in the server.

3.3 System Design

System design is a set of descriptions of the implemented software structure, the system data, and the components' interface and algorithms [67]. It is also considered to be a system model process. The most important activities in system design are architecture design, abstract specification, interface design, component design, data structure design, and algorithm design.

The application is suggested to be implemented based on Java, an object-oriented language. There are many important models for an object-oriented project design. One of them describes the static and dynamic relationships between objects. Another model explains how objects interact with each other.

In order to design this application, Unified Modelling Language (UML) is selected as the approach for the standardised specification notation with visualisation. UML has three different views of a system model, including functional requirements view, static structural view, and dynamic behaviour view [68]. The functional requirements view describes the requirements of the system from the user's viewpoint. The functional requirements and non-functional requirements of the application were discussed in Section 0 and Section 3.2.6. The static structural view uses objects, attributes, operations, and relationships to explain the system's static structure. The examples are class diagrams and composite structure diagrams. The dynamic behaviour view illustrates the system's dynamic behaviour among objects, and changes them to the

object's internal states. Sequence diagrams, activity diagrams, and state machine diagrams are all examples of this. In the project, two significant diagrams; a class diagram and a state diagram; are used for the system's static structure and dynamic behaviours.

3.3.1 Class Diagram

Sommerville [67] explained that an entity consisted of a state and that a defined group of operations operated on the state is called an object. The state contains a group of object attributes. Services to other clients are provided by operations associated with the object when requested. According to the object class definition, objects are created. An object definition is a type specification and a temple to create the object. It contains all attributes' and operations' declarations. In terms of the theory, there are eight classes defined in the application.

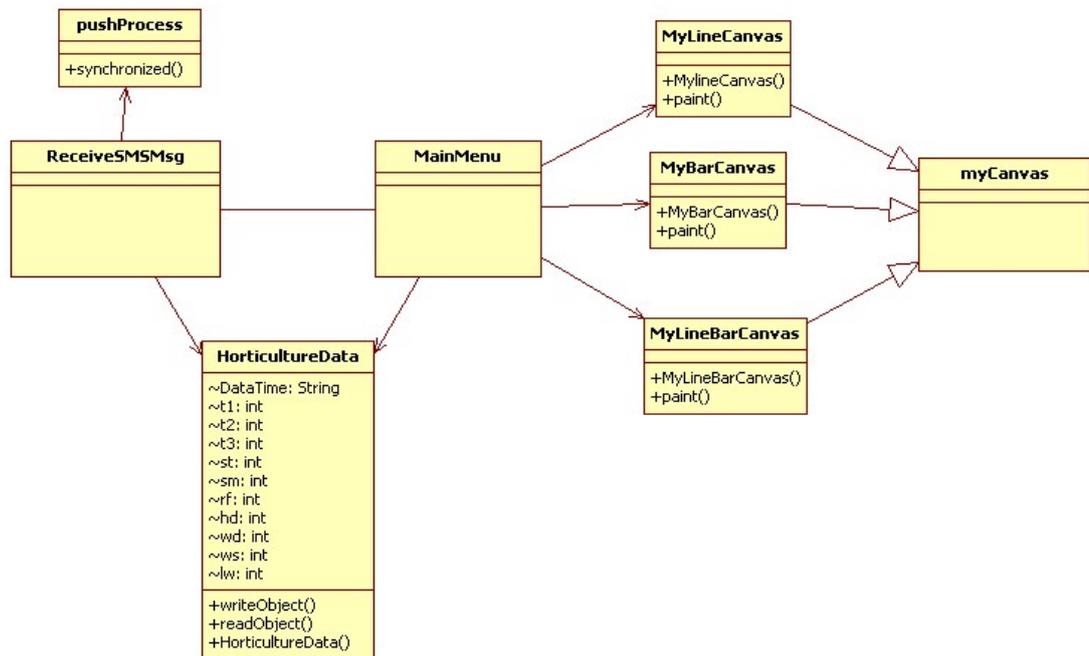


Figure 8: The Class Diagram for MicroBaseJ

Figure 8 shows the object class diagram for MicroBaseJ. This figure only contains the application of the mobile phone side and does not show the functions in the Sentinel and

the server. Usually, there is at least 200Kb of storage for each class when the application is deployed into a resource constrained device such as a mobile phone. To reduce the application size deployed in the mobile phone, only eight object classes are defined in this application; ReceiveSMSMsg, MainMenu, HorticultureData, pushProcess, myCanvas, MyLineCanvas, MyBarCanvas, and MyLineBarCanvas. Figure 8 also illustrates the attributes and operations for Horticulture and pushProcess, with other classes introduced in the following sections.

The class pushProcess deals with the message from the Sentinel. This object will be activated by the ReceiveSMSMsg object as a background process when the message arrives from the Sentinel. The HorticultureData is the class defined objects for the database. It has insert and read functions, and defines all attributes for the database, including time stamp and another ten items from the sensors. It is used for the ReceiveSMSMsg and the MainMenu classes.

The ReceiveSMSMsg is the main class in the application. It offers an entry to the application for the mobile user. It implements major functions of the application, such as receiving the message from the Sentinel, storing the relevant information into the mobile database, and providing the interface to the MainMenu class. Figure 9 illustrates the detail of the ReceiveSMSMsg class.

Figure 9 illustrates that the ReceiveSMSMsg has a number of attributes, such as the command buttons for the mobile user; NEXT_CMD, BACK_CMD, etc.; and parameters for the alarms of the ten items. The ReceiveSMSMsg also has a number of normal operations for mobile applications; for example, destroyed(), and startApp(). It has other operations for the function of receiving the SMS from the Sentinel, processing them, and storing them into the database. It also provides a connection to the MainMenu class and shows an alarm form on the mobile screen when the incoming reading exceeds the alarm limit.

```

ReceiveSMSMsg
~data: String
~done: boolean
+alarmFlag: boolean
~cmdExit: Command
~BACK_CMD: Command
~NEXT_CMD: Command
~SEND_CMD: Command
~SEARCH_CMD: Command
~alarmRow: String
~t1AlarmH: int
~t1AlarmL: int
~t2AlarmH: int
~t2AlarmL: int
~t3AlarmH: int
~t3AlarmL: int
~stAlarmH: int
~stAlarmL: int
~smAlarmH: int
~smAlarmL: int
~hdAlarmH: int
~hdAlarmL: int
~wsAlarmH: int
~wsAlarmL: int
~wdAlarmH: int
~wdAlarmL: int
~rfAlarmH: int
~rfAlarmL: int
~lwAlarmH: int
~lwAlarmL: int
~pushConn: MessageConnection
~listenerConn: MessageConnection
~MessageConnection: MessageConnection
~flagPush: boolean
~FlagPush: int
~interval: String
~lastDate: String
+Attribute1
~alarmRS: RecordStore
~sampleRS: RecordStore
~lastDateRS: RecordStore
~db: Storage
~root: Index
~HData: HorticultureData
~PAGE_POOL_SIZE: int
~mainMenu: List
~alarmForm: Form
~alert: Alert
~p: Player
~output: String

~error()
+ReceiveSMSMsg()
#startApp()
~handlePushActivation()
~addAlarmRecord()
~startupDisplayData()
~pushDisplayData()
~DataRound()
~DataStore()
#pauseApp()
#destroyApp()
~quit()
~close()
~AlarmForm()
+commandAction()
+exitMIDlet()
+notifyIncomingMessage()
+msgProcess()

```

Figure 9: The Illustration for the ReceiveSMSMsg Class

The MainMenu provides an interactive interface to the user to manipulate relevant operations. It allows the user to choose different functions, such as presenting the user with specific information on the screen, updating the alarm settings and sample rates, and sending the modified sample rate back to the Sentinel. It also provides an entry to one of the MyLineCanvas, MyBarCanvas, and MyLineBarCanvas.

Figure 10 shows that the MainMenu contains several attributes, such as the alarms, units, forms, choicegroups, and textfields for user interfaces. The MainMenu also has a

number of operations. For example, the welcomeMenu() provides a splash form to the mobile user.



Figure 10: The Illustration for the MainMenu class

The class myCanvas is an abstract class, which plots the relevant data retrieved from the database on the mobile screen. It defines the basic chart for the mobile application.

Figure 11 illustrates the class myCanvas. It defines the necessary attributes for the graphs. It also defines four operations for painting, keyPressed, keyReleased, and myCanvas itself.



Figure 11: The Illustration for the myCanvas Class

The classes; MyLineCanvas, MyBarCanvas, and MyLineBarCanvas; are inherited from the class myCanvas. They are used for drawing different types of figures, such as the line chart, the bar chart, and their combination.

3.3.2 State Diagram

The Statechart diagram is one of state machine models in UML, with state machine models being dynamic models that describe the actions in the software engineering process.

As can be seen from Section 3.3.1, there are eight classes in the application. The most important classes are the ReceivesSMSMsg class, the interface classes, MainMenu, and the three canvases. These classes' state diagrams are analysed in the following section

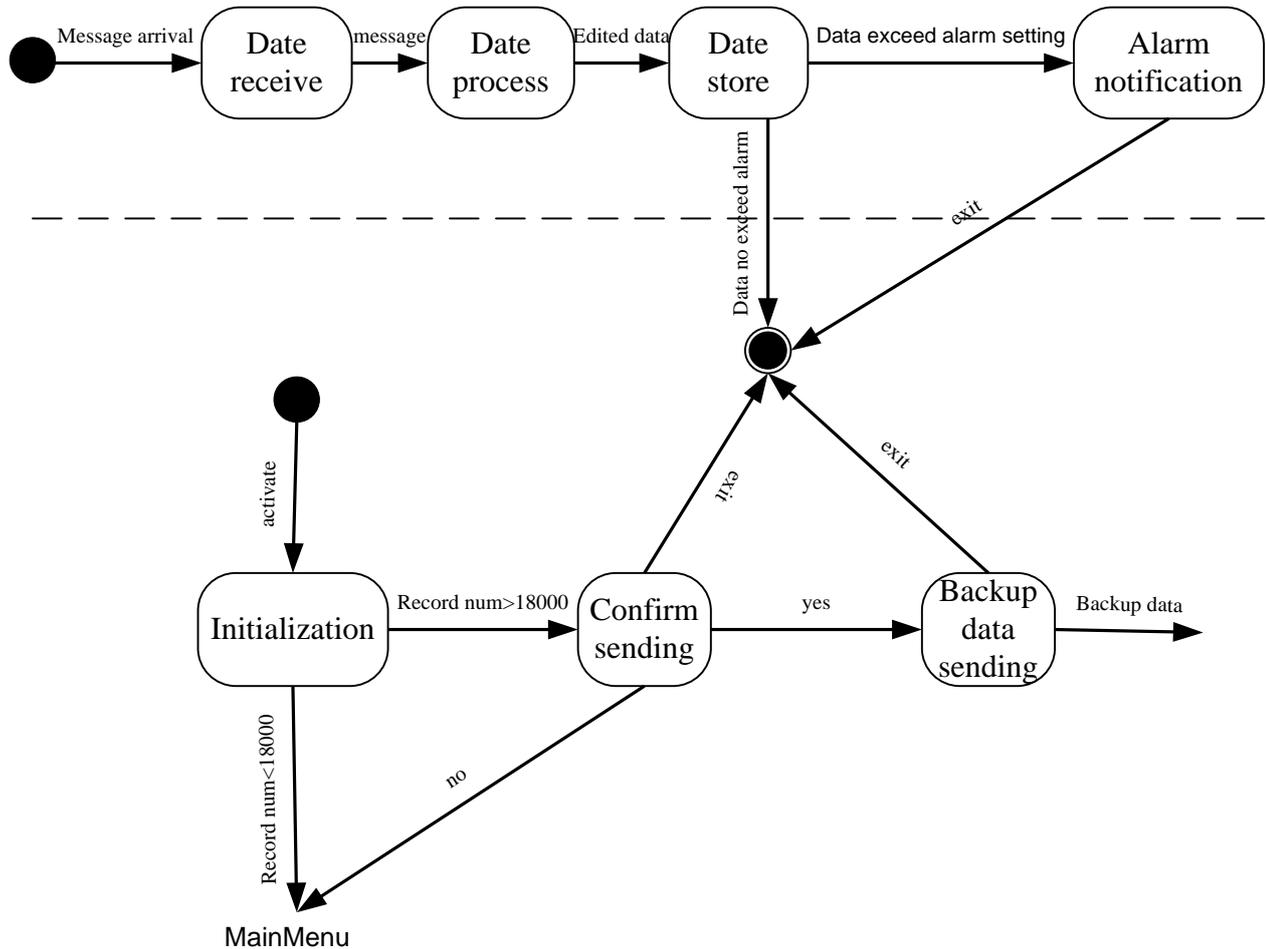


Figure 12: The Statechart Diagram for the ReceiveSMSMsg class

Figure 12 shows the statechart for ReceiveSMSMsg. It presents the major states in the class, ReceiveSMSMsg. The object has two entries, it can be activated by the mobile user, as well as by a message from the Sentinel. The SMS from the Sentinel is processed as background if no items exceed the alarm limits. Otherwise, an alert form is shown on the mobile screen.

No matter whether the application is activated by the user, or by a message from the Sentinel, once a new message arrives, the object ReceiveSMSMsg receives it and processes it as a particular format, and then stores it into the object -- Horticulture. The ReceiveSMSMsg also checks whether the items' value exceeds the alarm settings. If so, an alarm form is shown on the mobile screen. Otherwise, the application will be terminated if it has been activated by a message, or the application keeps running normally if it has been activated by the user.

When the application is activated by the user, it will check whether the record number of the embedded database exceeds 18,000 records. If so, then a notification is shown on the mobile screen to allow the user to choose to backup the data, exit the application, or continue it. If the backup function is chosen, the application asks the user to set up the Bluetooth connection to the server and then send the first-in 5,000 records to the server. Finally, the 5,000 records are deleted from the mobile database. If exit is selected, the application will be terminated. The application calls the MainMenu if the user chooses to continue.

Figure 13 illustrates the MainMenu class and the three canvases. It presents the major states in the class MainMenu and the inherent class, myCanvas. This object is called the ReceiveSMSMsg. The object deals with the major interfaces with the user. It provides interfaces to users and presents the data charting, the alarm, and the sample rate configuration. The updated alarm setting and sample rate are stored into the files in the mobile, and the updated sample rate is sent to the Sentinel for next sampling.

The MainMenu provides four options for the user, including charting, alarm configuration, sample rate configuration, and exit. The charting process provides several cascade menus for the user, including an item choice menu, date choice menu, the form for maximum and minimum values, and the data plotting for a specific date. Each menu provides two choice buttons, next and back, to allow the user to continue, or return to the previous menu. The chart item choice menu also checks the user chosen (limited to two items). While the retrieved date is typed in, the object checks its validity and searches the specific data from the mobile database. A form identified the max and min value in the specific date comes on screen after the user presses the Next button. Finally, the line, or bar, charts are shown on the screen when the subscriber pressed the

Next button. The user interacts with the data using the 2, 4, 6, and 8 keys of the mobile phone.

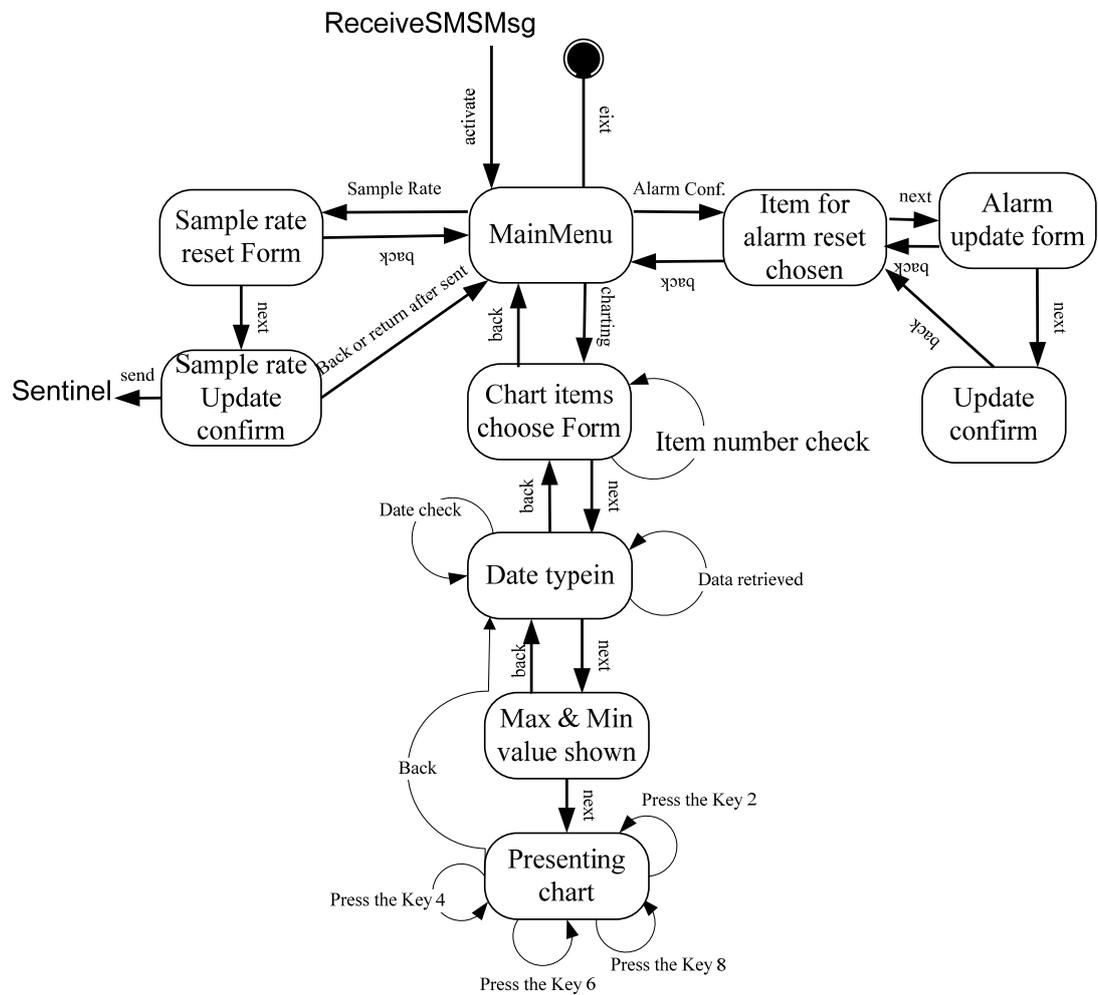


Figure 13: Statechart Diagram for the MainMenu class

The alarm configuration provides an item choice menu, and an updating form. Once the alarm is updated, a confirm form is provided for the user. The sample rate configuration allows the user to update the current parameters and provides a confirm form as well. The updated sample rate is sent to the Sentinel after the confirmation.

3.4 Summary

This chapter, system design, is the crucial element in the project. MicroBaseJ is designed using software engineering concepts. It covers all features for system design in software engineering.

This chapter analyses the requirements for this project, including the functional and non-functional requirements. User requirements are collected and analysed, and then the general architecture for the application is proposed. The chapter also discusses data sources for the application; including four temperature values, humidity, soil moisture, wind direction and speed, rainfall, and leaf wetness. Next, the communication protocols between the Sentinel and the mobile phone are proposed to adopt SMS, and Bluetooth is suggested for the communication between the mobile phone and the backup server. The interface description, data Structure between components and dataflow model for the project are also discussed. MicroBaseJ roadmap is also discussed.

The class diagram of MicroBaseJ is discussed, including the eight classes ReceiveSMSMsg, MainMenu, HorticultureData, pushProcess, myCanvas, MyLineCanvas, MyBarCanvas, and MyLineBarCanvas. The details of the ReceiveSMSMsg, MainMenu, and myCanvas classes are also investigated. The state diagrams of ReceiveSMSMsg, MainMenu, and myCanvas are discussed.

4 Mobile Phone Database and User Interface Implementation

4.1 Introduction

This chapter investigates the implementation of the mobile database and the user interfaces for MicroBaseJ. It discusses the data format and data storage for the database. Next, the design of the user interfaces is discussed in this chapter. The hardware and software for the user interfaces are investigated. This chapter also discusses the user interface design from the screen layout, command, and menu for the application.

4.2 Database Design

4.2.1 Data Format

As mentioned in Section 3.2.2, the data sources of the project include four temperature readings, wind speed, wind direction, humidity, soil moisture content, rainfall, and leaf wetness. The details of these items have been fully discussed in Section 3.2.2. The ten items are collected from sensors in the Sentinel and attached with a time stamp. Table 9 describes the eleven items' format, data range, unit, size, and comment.

The time stamp is given as ten digit number. The data format of the time stamp is the string type. The time stamp format is defined as: ddmmyyhhmm; where dd stands for day, mm is month, yy represents year, hh is for hour, and mm is minute.

The format of the four temperature values is set as the Integer type. The value of the temperature values vary from -50 to 125. The formats for humidity and soil moisture are both defined as the Integer type. Their ranges vary from 0% to 100%. The wind speed format is also defined as the Integer type, its value varying from 0 to 200 km per hour. The wind direction adopts the Integer type as its format. It has 16 values which represents 16 different directions of the wind.

Table 9 - Database format for MicroBaseJ

Items	data type	data format	range	Unit	Comment	bytes
Time Stamp	string	ddmmyyhhnn (dd: day, mm: month, yy: year; hh: hour, nn: minute)	dd: 1~31, mm:1~12, yy:00~99, hh:1~24, mm:1~59		ID for each sample data	10
t1	integer		-55 ~ +125	°C	The first temperature	4
t2	integer		-55 ~ +125	°C	The second temperature	4
t3	integer		-55 ~ +125	°C	The third temperature	4
st	integer		-50 ~ +50	°C	soil temperature	4
sm	integer		0 ~ 100	%	soil moisture	4
rf	integer		0 ~ 99	mm	rainfall	4
hd	integer		0 ~ 100	%	humidity	4
wd	integer		1~16 (1: N, 2: NNW, 3: NW, 4: NWW, 5: W, 6: WWS, 7: WS, 8: WSS, 9: S, 10: SSE, 11: SE, 12: SEE, 13: E, 14: EEN, 15: EN, 16: ENN		wind direction	4
ws	integer		0 ~ 200	km/h	wind speed	4
lw	integer		w: wet; d:dry		leaf wetness	4

The rainfall's format is defined as the Integer type. Its unit is millimetre and its value from the sensor in MicroBaseJ varies from 0 to 9,999 mm. The format of the leaf wetness is also defined as the Integer type. It only has two values; 0 or 1; where 0 stands for dryness in the surface of the leaf, while 1 stands for wetness. The Integer type in Java only occupies 32 bits, while the Double type needs 64 bits. Thus, all items except the time stamp are set as the Integer type, rather than the Double type.

4.2.2 Data Storage

There are more than several thousand records in MicroBaseJ. How to manage the storage and manipulation of records becomes the challenge of the project. Usually, for mobile applications, there are two ways to store records. One way is to store records in the server. The mobile user retrieves data, which are stored in the server from the

handset. This requires wireless Internet, or a specific connection between them. Another way is to store records into the mobile phone itself. The mobile phone has been considered as a resource-constrained device for many years. Nevertheless, with hardware prices going down significantly, the mobile phone has become more powerful than ever. Nowadays, the mobile phone is programmable, supports various OSs (operating systems), and has large sized memory of over 1GB. Hence, data management in a mobile phone is now realisable.

Several embedded databases have been developed for PDAs, such as db4o and PointBase, whereas few focus on mobile phones. J2ME provides a file management system – RMS for the data persistent management of the mobile phone. RMS stores and manipulates data as records. One drawback of RMS is that it only provides a unique ID number, rather than an index for data retrieval.

Apart from RMS, Perst Lite is one of the embedded databases available for limited-resource devices. Perst Lite is an open source object-oriented database, which was developed by McObject [40]. The object-oriented database can be seamlessly integrated with object-oriented programming languages, such as JAVA, and renders the development to be simpler, as well as enhancing the performance of limited-resource devices, such as mobile phones.

Perst Lite is chosen as the database for MicroBaseJ. It is very easy to use, with high performance. It stores and fetches data as objects and provides an index for the key retrieve. Perst Lite does not like the relation database, and it can be integrated tightly with a programming language, such as J2ME, because it is seamless with the application.

One advantage of the database is the index. Using the index for data retrieval improves the search performance. For the database, there are two methods to enhance the performance; increasing the page pool size and the index. In MicroBaseJ, the page pool size is set 64KB, and it improves the performance by reducing the access time for data storage, or retrieval. The other technique is to set the time stamp of the record as an

Table 10 - The size comparison of RMS & Perst Lite

Record numbers(K)	Perst Lite(KB)	RMS(KB)
-------------------	----------------	---------

1	282	111
2	498	222
3	764	333
4	980	445
5	1225	556
6	1635	667
7	1848	778
8	2073	890
9	2314	1001
10	2530	1112
11	2755	1228
12	3378	1340
13	3586	1452
14	3803	1565
15	4024	1677
16	4269	1790
17	4486	1902
18	4707	2015
19	4924	2128
20	5141	2241

index. After storing the record, the time stamp is inserted into the index. The user types in the particular time with the day, month, and year, in order to locate the relevant records quickly when retrieving them.

Each database needs a DBMS (database management system), however, which requires extra expense, in terms of additional space and complexity. Table 10 compares the size of RMS with Perst Lite based on various record numbers. The record format is discussed in Section 5.2. As can be seen, the size of Perst Lite is twice that of RMS for the same record numbers.

4.3 User Interface Design

How to design the user interface efficiently, attractively, and interactively, are among the important issues in the project. The basic idea of the user interface design in MicroBaseJ is that user interfaces should be controllable, scalable, flexible, portable, and affordable. The user interface design covers the hardware and software chosen, and the interface design itself.

4.3.1 Hardware for User Interface

The devices for MicroBaseJ require at least a 2.5G or 3G mobile phone, with MIDP2.0 and CDLC 1.1. The mobile phones provide the user interfaces to users to facilitate interaction with the data stored in the mobile embedded database in MicroBaseJ. Cascaded menus are provided to users and the results are presented as a line, or bar chart.

4.3.1.1 Screen Size

Appendix I compares a number of the models, technology, and screen sizes for major brand mobile phones. All of them support MIDP 2.0 and CDLC 1.1. As can be seen, there are four main brands in this table: Motorola, Nokia, Siemens, and Sony Ericsson. Each brand contains various models. The result shows that various brand mobile phones have different resolutions. Screen sizes vary across the various brands as well. Even though they are of the same brand and are approximal serials, their resolutions are still varied. For example, the Nokia 3220's resolution is 128 x 128 pixels, whereas the Nokia 3230's screen size is 176 x 208 pixels, although they are in the same Nokia 3200 series.

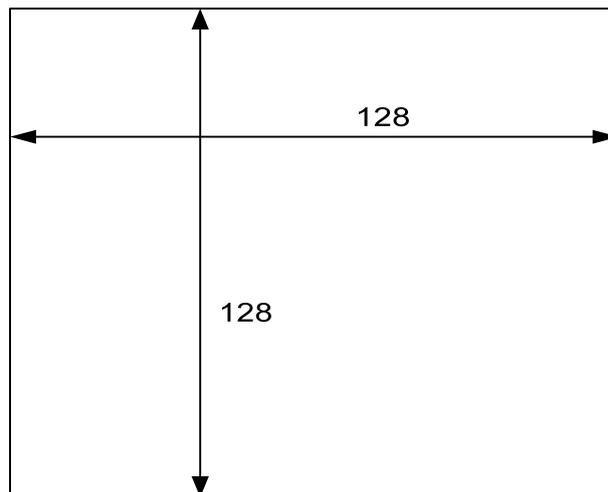


Figure 14: The minimum resolution for mobile phones

Appendix I shows that the horizontal pixels in these phones vary from 128 to 352 pixels, while the vertical pixels vary from 128 to 420. From this, it can be seen that it is very important to consider how to fit interfaces to various mobile screens when the line chart and bar chart are designed.

Suppose a minimum size, 128 x 128 pixels, as a general solution for mobile phones. Figure 14 shows the supposed minimum resolution for mobile phones.

4.3.1.2 Other devices

There are many devices for the user interface interaction, such as the monitor, the keyboard, the mouse, the touch screen, and the joystick. In the case of the mobile phone, however, the limitations are obvious; it only has a small screen and twelve keys, 0~9, * and #, plus several functional keys. Perhaps a few up-to-date models provide a touch screen for the user. However, the number of these phones is limited and they are relatively expensive.

Due to the resource constraints, mobile phones for the project should be affordable, not expensive. For the average customer, devices must be focused on common cell phones, rather than the more powerful devices available in the market.

To allow users to interact with the data presented in the chart, a yellow trace ball is designed for the project. The yellow trace ball is set as a small, round ball to allow users to interact with data points. Users use the key 6 to forward the trace ball to the next data value, whereas the key 4 allows movement backwards to the previous data point. At the same time, the trace ball can be swapped between the two charts by using the key 2, and the key 8. The key 2 is for the upper chart, while the key 8 is set for the bottom figure.

4.3.2 Software Chosen for User Interface

J2ME, the developing language in the application, provides a GUI API for the programmer, including the low-level API, Canvas, and the high-level API, Screens. The Screen contains Form, List, Alert, and TextBox subclasses. The last three items are the predefined components. The former is the open type and it is a container to support

multiple items. The programmer uses the low-level API, Canvas's paint method, to draw the screen picture.

The high level API helps the programmer develop the user interface to be fast and portable. The programmer cannot, however, change the presentation of the GUI. For example, it does not allow the user to define the sharpness, colour, and font. Although the programmer can use a low-level GUI API to generate an attractive interface, it needs the programmer to work on much manual porting to allow the user interface to be run on all, or most, J2ME mobile devices.

The free third party software J2ME Polish was introduced by Virkus [58]. J2ME Polish provides a user interface and build tools. The J2ME Polish GUI is not only compatible with the high level GUI API, but also allows the programmer to design the detail of the user interface. The J2ME Polish has a number of unique characteristics, including easy implementation, automatic porting, innovative designs, customisation, flexibility, and extensibility.

In MicroBaseJ, the J2ME Polish GUI is selected to design interfaces except for the data plotting. It has its disadvantages, however, such as the need for extra space in the JAR file. Concerning charting, J2ME Polish only provides a line chart. The flexibility of labels on the x and y axes are limited. Another key disadvantage is that the user cannot interact with the data in the charts.

Thus, a customer class; to illustrate the data from the embedded database as a chart; is proposed. The customer class allows the user to interact with the user interface using a yellow tracing ball. The tracing ball is controlled by the keys in the mobile phone panel. The class is extended from the low level API, Canvas.

4.3.3 User Interface Design

To design a user interface, there are a number of features that need to be considered, including simplicity, visibility, and consistency. To implement a good-looking user interface, the screen layout, commands, and menus need to be well-organised.

4.3.3.1 Screen Layouts

In the screen layout, the screen arrangement, colour, and graphing structure are very important features in the user interface design. As eyes usually move from left to right, the charts in MicroBaseJ are presented along the horizontal direction. Data points are also organised from left to right, according to their time sequence.

The background of the user interfaces is set as white, because a white background presents the chart and menu clearly. Meanwhile, a number of specific icons are used for the identify of particular items; for example, a figure of the bar chart represents the plotting function, the alarm configuration uses the tools figure, a figure of a temperature measurement represents temperature, and the figure of a cloud with rain represents rainfall.

Table 11 illustrates the relation of items on the main menu and their icons. Table 12 shows the corresponding relation between the data sources and their icons.

Table 11: The Relation of Main Menu Items and Icons

Items	Icon	Items	Icon
Charting		Alarm Configuration	
Sample Rate Configuration		Exit	

Table 12 - The Relation of Data Source and Icon

Data sources	Icons	Data sources	Icons
temperature1		soil moisture	
temperature2		wind direction	
temperature3		wind speed	
soil temperature		rainfall	

humidity		leaf wetness	
----------	---	--------------	---

The background colour for icons, including temperature, humidity, soil moisture, and wind speed, is set as light blue, because it attracts the user and the user feels comfortable with the colour.

4.3.3.2 Commands and Menus

Due to the small screen of the mobile phone, a cascade menu is proposed for MicroBaseJ. According to the 7 ± 2 rule [48], the number of chunks on the mobile screen are set to be less than 9. The user presses the multi, or exclude, choice button to select the specific item.

The Next button, Back button, and Exit button are provided for the interaction operations in the user interfaces. The main menu provides the Next button and the Exit button, whereas the chart menu only displays the Back button. Other interfaces use the Back button to exit from the current, and back to the previous, menu, while the Next button is used for entry to the next menu. In case of an emergency, the application can be stopped immediately by pressing the Exit functional key on the mobile phone.

The application allows the user to enter the specific date to retrieve the data from the embedded database. To avoid the entry of too many digits by the user, the input field is defined as a specific field. The date of the recent stored data is set as the default for the input field. The entered data and commands are checked for validity instantly.

After choosing the particular item, the user presses the Next Button to enter the next menu. The user presses the same keys for similar functions on different menus during the whole process. The application also provides a confirmation button for each parameter update, including the sample rate and the configuration update.

To identify the alarm data, its colour is set as red on the text description at the top of the mobile screen, while the colour for the normal data is presented in black.

4.3.3.3 The Design for the Graphic User Interface

An application with a good-looking and responsive user interface is considered to be professional and excellent [58]. Thus, the graphic user interface for data plotting is the most crucial element in the application.

To illustrate the data on the small various mobile phone screens is one of the challenges in the project. As discussed in Section 4.3.1.1, the resolutions of mobile phones are noticeably variable. To deal with the problem, a minimum resolution, 128 x 128 pixels, is assumed in the research. The design for the graphic user interfaces is based on this minimum resolution.

On the horizontal direction, 20 pixels are used for labels and axes to identify the data values on the left, and as well as 5 pixels being used on the right. Thus, there are only 103 pixels remaining for the chart. When drawing the line chart, each data point needs to use a pixel to identify its value. For a bar chart, each bar needs at least two pixels to present the data point's value. Therefore, the maximum points for the bar chart are 51 ($(128 - 20 - 5) / 2 = 51.5$). Suppose the sample rate is per hour. Under this assumption, each day has 24 samples. Thus, the ideal data points for each bar chart are limited to 48 points for one graph. It is supposed that either the bar chart, or the line chart, has a maximum of 48 data points in one figure to unify them.

On the other hand, from the vertical viewpoint, 128 pixels is the minimum size assumed for the mobile phone resolution. Fifty pixels are used for the illustration of the date, time, and the value for the cursor data point on the top of the screen. To manage the data efficiently, two charts are bundled on the mobile screen each time. Suppose that there is a gap of 15 pixels used for distinguish the two figures. Thus, the minimum height of each figure is 31 pixels ($(128 - 50 - 15) / 2 = 31.5$).

Figure 15 shows the graphical user interface design for the data plot. As can be seen, the pixels for label, titles, and the cursor point's value and gap are fixed. The remaining pixels for the first and second figures can be variable, according to the various mobile resolutions.

The vertical pixels range from 128 to 420 and the horizontal pixels range from 128 to 352, in differing brands and models. Thus, the scalable chart is a challenge to the application designer. Another key point is that the value of the data sources varies from -55 to 999.

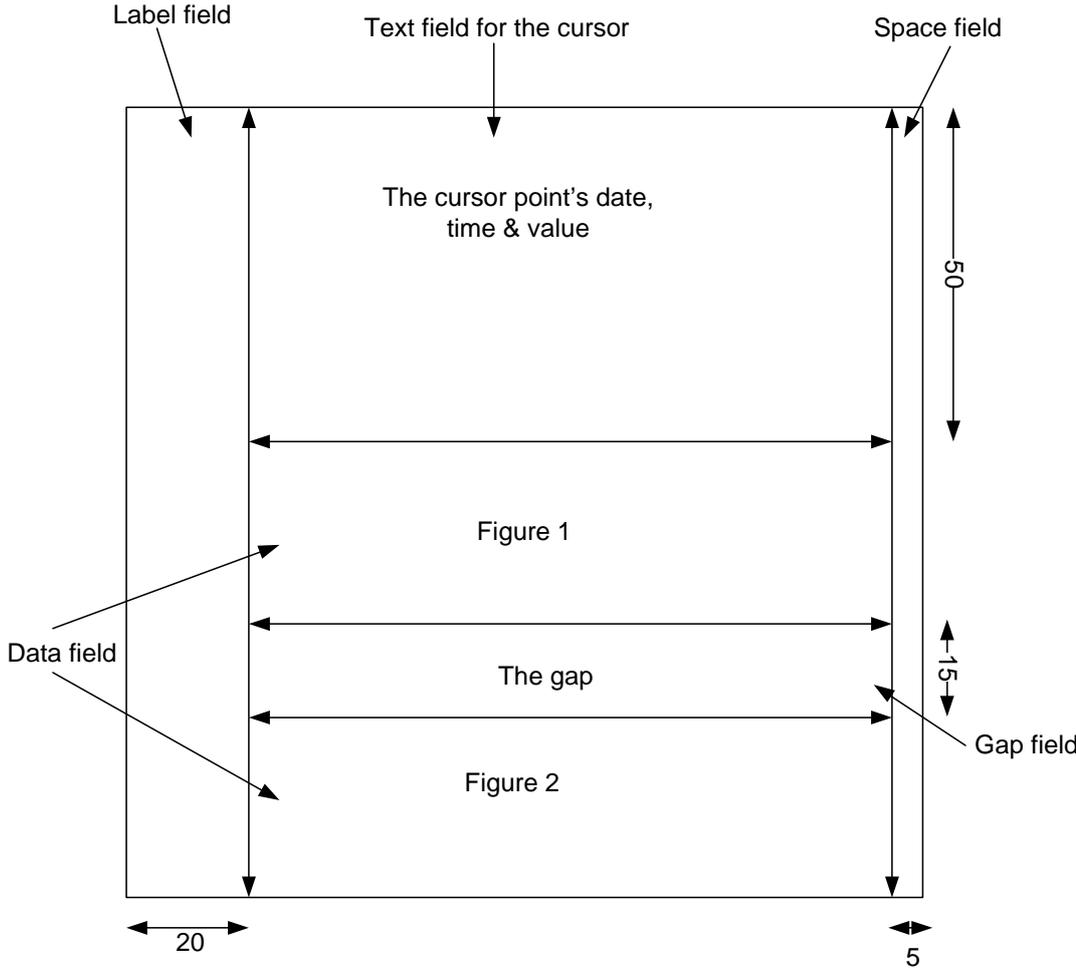


Figure 15: The Mobile User Interface Design for Chart

In MicroBaseJ, charts are designed to be very flexible and adjustable to various screen sizes. As mentioned above, the maximum points in each graph is 48. These points are distributed on the screen according to the currently available screen size. From the horizontal direction, the design principle is more pixels on screen, and more spacing between each point. This means that the chart is adjusted to

various resolutions. Regarding the vertical axis, the height of the chart can be scalable to a variety of screens according to the maximum and minimum values of the items, and the resolution. For example, the more screen pixels there are in the height, the more the chart is enlarged, while a shorter screen means that the figure is shortened to fit into the screen.

Figure 16 illustrates one of the customer graphic interfaces based on the WTK 2.5.1 default ColorPhone simulation. As can be seen, a yellow traceball is provided to the user to interact with the data. The interface can be divided into three parts, including the text field and other two figures. There is a gap between each figure. On the left of each figure, several labels are used for identifying the relevant axes.

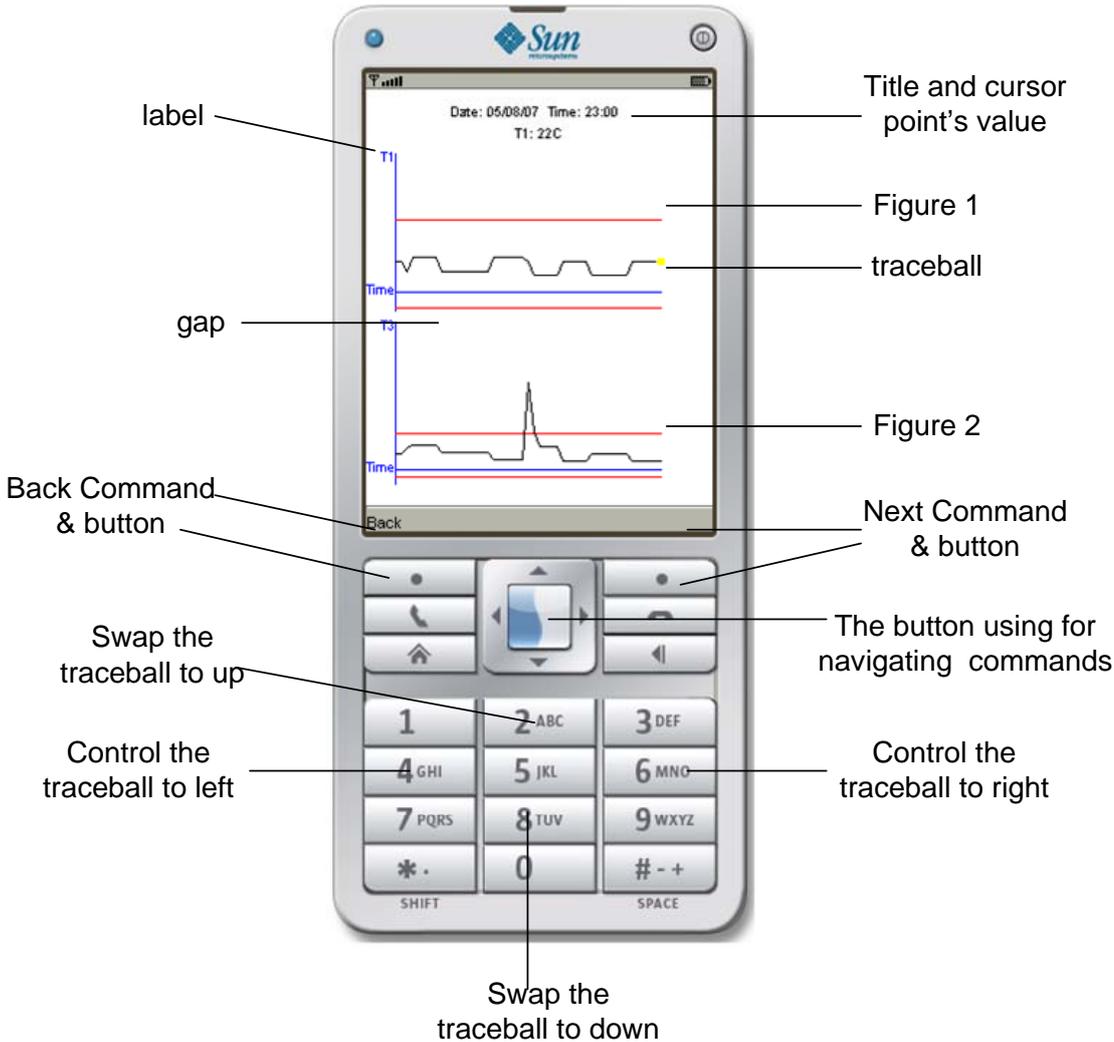


Figure 16: The Illustration of the Graphic Interface

As mentioned in the previous section, the title, gap, and labels are fixed. The width and height of two figures can, however, vary according to the resolution of the mobile phone.

The text field shows the date, time, and value of the cursor point on the screen top. If the value exceeds the alarm limit, the text colour is set at red, otherwise black text is shown on the top.

There is a Back button at the left bottom of the screen. The user presses the Key under the icon to return back to the previous menu. The Key 2 and Key 8 are used for swapping the yellow traceball between the two figures. The Key 4 and Key 6 are designed to control the yellow traceball to the left and right. The detail of the menu and operation is given in Appendix II.

4.4 Summary

This chapter discusses the implementation and design of the embedded database and the user interfaces for MicroBaseJ. It also discusses the data format and data storage for the embedded database. There are eleven items adopted for the application; including time stamp, temperature readings, humidity, soil moisture, wind direction and speed, rainfall, and leaf wetness. The application adopts the object-oriented database called Perst Lite, as the database embedded into the mobile phone. The time stamp and the index are set as String type formats. The other attributes' formats are set as the Integer type.

This chapter also investigates the user interfaces for MicroBaseJ. The hardware and software for the user interface are discussed. A yellow traceball is proposed for the user to interact with the data in the data plot. A third party software, J2ME Polish, was introduced into the graphic user interface design. This chapter also discusses the screen layout, command, and menu for user interfaces in the application. Finally, the customised user interface is investigated based on the minimum resolution of 128 x 128 pixels. In general, the customised user interfaces are designed to be controllable, scalable, flexible, and portable.

5 Evaluation and Discussion

5.1 Introduction

This chapter describes the performance studies and results obtained for the embedded database and MicroBaseJ. It discusses the embedded database performance and MicroBaseJ's performance issues. Firstly, the performance of Perst Lite was compared with RMS, based on both the simulation and the real device. The performance evaluation covers the insert function, sequence search function, and random search function. The recorded numbers for these testings varies from 1,000 to 20,000, with 1,000 records' placed in each evaluation. Their storage sizes were also compared. Next, the application performance was investigated. The evaluation for this application focuses on the mobile phone side. The record number for the application ranges from 1,000 to 18,000 at each thousand records. The performance was evaluated according to the application initialisation, data retrieval, data selection, and data plotting.

5.2 Database Performance Evaluation (Perst Lite & RMS)

Choosing an appropriate embedded database is one of the objectives in the project. To obtain a satisfactory result, it is necessary to evaluation the performance of embedded databases. There are several essential operations for a database, such as the insert function, search function, delete function, and modify function. The insert function is used for adding a new record into the database. The search function focuses on quickly locating the specific record. The delete function is used for cancelling the specific record. The modify function allows the user to update the particular data. From above, it is obvious that the behaviour of the search function, the modify function, and the delete function are similar. Their operations are based on locating the specific record. Thus, of them, only the search function and the insert function need to be evaluated for the embedded database. The search function of the database includes a sequence search and a random search.

Two embedded databases were evaluated, Perst Lite and RMS, which were discussed in Section 1.4. The performance of the insert function, the sequence search function, and the random search function of Perst Lite and RMS were evaluated. These three functions were assessed in both the simulation and the real device. The data range for the experiments varied from 1,000 to 20,000 and the evaluations were implemented at each thousand records. The Perst Lite pool size is set as 64K, 100 records being one transaction. The outcomes were the average executing time for each record, in each operation. The outcomes were the average executing time for each record, in each operation.

There were 20,000 records used for the testing of these operations. Each record contains several items and the total size is 111 bytes, including the String and Integer type attributes. One of them is used for identifying the record order. In Perst Lite evaluation, it is set as an index for the fast location of the specific record through the use of a random generator.

The experiments were developed in J2ME with J2ME Polish and based on the Java platform jre1.6.0_01 and the Eclipse DSK 3.2.2.

5.2.1 Simulation

The embedded databases' performance of Perst Lite and RMS was estimated on the mobile simulation, Sun Java Wireless Toolkit 2.5 DefaultColorPhone.

5.2.1.1 Insert Function

The executing time only measures the exact insert time for each insert operation, rather than for the whole application. The outcome is the average time for the insert operation of each record, at each assessment, and its unit is msec.

Figure 17 compares the performance of Perst Lite and RMS on the insert function. It can be seen that the time complexity for Perst and RMS are both close to $O(1)$. The reason for this is that RMS inserts the new record at the last, and Perst Lite uses an index. It is interesting to observe that Perst Lite uses around one fifth of the running time of RMS for the insert function in each assessment. The major reason for its better performance is that Perst Lite uses 100 records per transaction, and this can greatly reduce the I/O overhead.

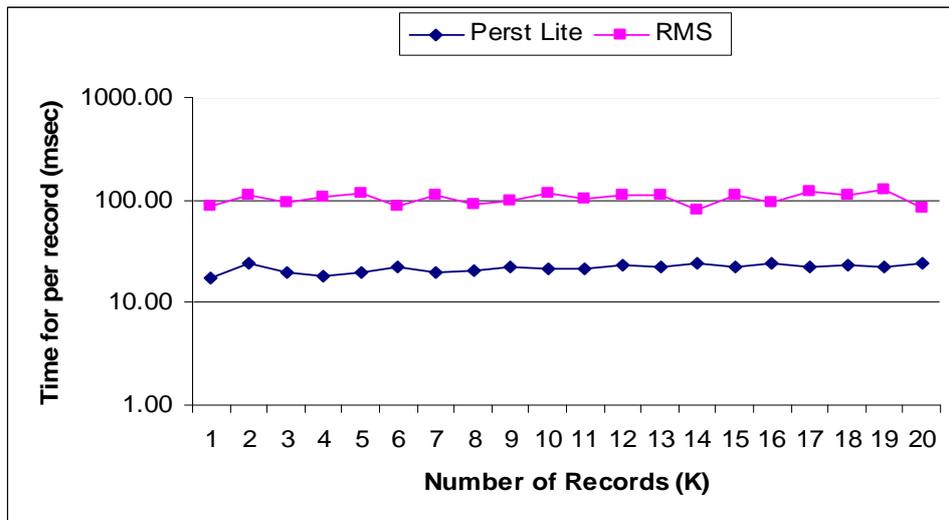


Figure 17: The Insert Function Comparison on Simulation

5.2.1.2 Sequence Search Function

The sequence search function includes the operation to fetch all records into a container, with all of them retrieved. RMS puts all records into the enumerator and searches them from the beginning to the end. Perst Lite sets all records into the iterator and retrieves them. The execution time is counted for the search operation's time. The outcome is the average time for the sequence search operation of each record at each assessment and its unit is measured in msec.

The performance comparisons for Perst Lite and RMS in the sequence search function are shown in Figure 18. The result shows that the performance of Perst Lite is better than that for RMS in each assessment. It is interesting to note that the execution time of the sequence search function for RMS increases noticeably with the increase of the record number, whereas Perst Lite remains stable if the record number is less than 16,000, however, the search execution time rises significantly after that.

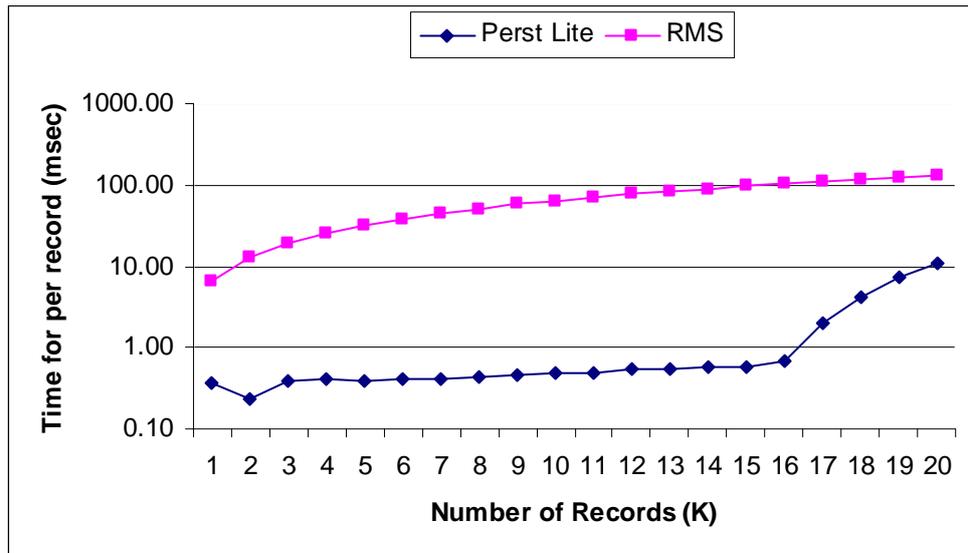


Figure 18: The Sequence Search Comparison of Perst Lite and RMS

5.2.1.3 Random Search Function

Random search is the important function for the project, because the user usually types the specific date to search for the relevant data in the database. The execution time is referred to as the exact operation’s time for the specific date searched from the database. The outcome is the average time for the random search operation of each record at each assessment, with its unit measured in msec.

As mentioned in Section 5.2, a number of random numbers were generated in this assessment and they were used for locating the specific record in the database. Firstly, 1,000 random numbers were used for the evaluation of Perst Lite and RMS. It is observed, however, that it took more than one day to implement 100 random searches in RMS when the number of records exceeded 13,000. It is obvious that the RMS performance of the random search function is significantly slower than the Perst Lite performance, either with an index, or without an index. Thus, the random number for RMS is reduced to 100 in each assessment. The outcome for RMS is the average time for 100 searches at each assessment. The outcomes for Perst Lite with an index, and without an index, are the average time for 1,000 searches at each assessment.

The random search function of Perst Lite is set in two ways, Perst Lite with an index and Perst Lite without an index. Perst Lite with an index uses a random number to match the index for the search function. Perst Lite without an index stores the object into a container and matches each record in the container with the random number. Similarly, RMS puts all records into a container and each record is matched with the random number.

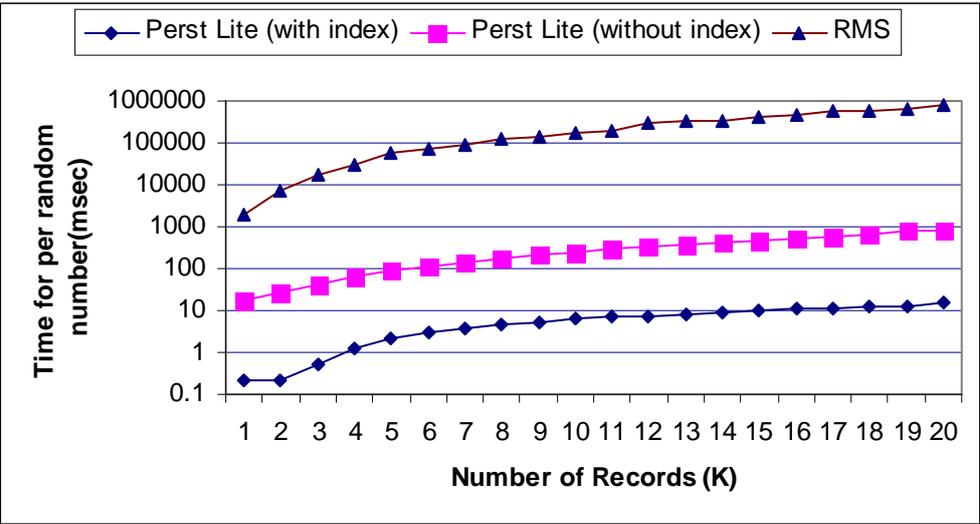


Figure 19: Random Search Comparison for Perst Lite with and without index and RMS

Figure 19 illustrates the comparisons of the random search function for RMS, Perst Lite with and without index. It is obvious that the random search of RMS increases drastically in length. The execution time of RMS for 20,000 records is 390 times that for 1,000 records. The random searches of Perst Lite with an index and Perst Lite without an index also increased markedly with increases in the record number. The execution time for the maximum record number is 69 times that of the minimum record number in the random search function for Perst Lite with an index, while the running time of the maximum record number is 47 times that of the minimum one in the random search function for Perst Lite without an index. It is observed that the random search function performance for Perst Lite with an index and without an index shows a significantly better result than does

RMS. The reason for this is that Perst Lite is set at 64K for the pool size, which helps to decrease the I/O operation. Furthermore, the random search function without an index takes 10 times the time of the one with an index. The Perst Lite random search function with an index takes less time because the index helps the user to locate the specific record more quickly.

5.2.1.4 Database Size Comparison

Table 13 compares the database size of Perst Lite with RMS based on the Sun Java Wireless Toolkit 2.5 DefaultColorPhone simulation. The record length is about 110 bytes. It is interesting to observe that Perst Lite needs nearly 2.5 times the storage space of RMS. This means that Perst Lite needs around 160 bytes extra in each record for the database overhead.

Table 13: The Database Size Comparison of Perst Lite and RMS

Record numbers(K)	Perst Lite(KB)	RMS(KB)
1	282	111
2	498	222
3	764	333
4	980	445
5	1225	556
6	1635	667
7	1848	778
8	2073	890
9	2314	1001
10	2530	1112
11	2755	1228
12	3378	1340
13	3586	1452
14	3803	1565
15	4024	1677
16	4269	1790
17	4486	1902
18	4707	2015
19	4924	2128
20	5141	2241

5.2.2 Real Device

The following estimations for the embedded database were evaluated on a real mobile phone; a Nokia N73 with 42M memory and an extra 1G miniSD card. Similar with the simulation, the assessment includes the performance of the insert function, the sequence search function, and the random search function of Perst Lite and RMS. The evaluations of the mobile phone testing are similar to that of the simulation. The Perst Lite pool size is also set as 64K, with 100 records as one transaction. The data range for the experiments varied from 1,000 to 20,000, and the evaluations were implemented at each thousand records.

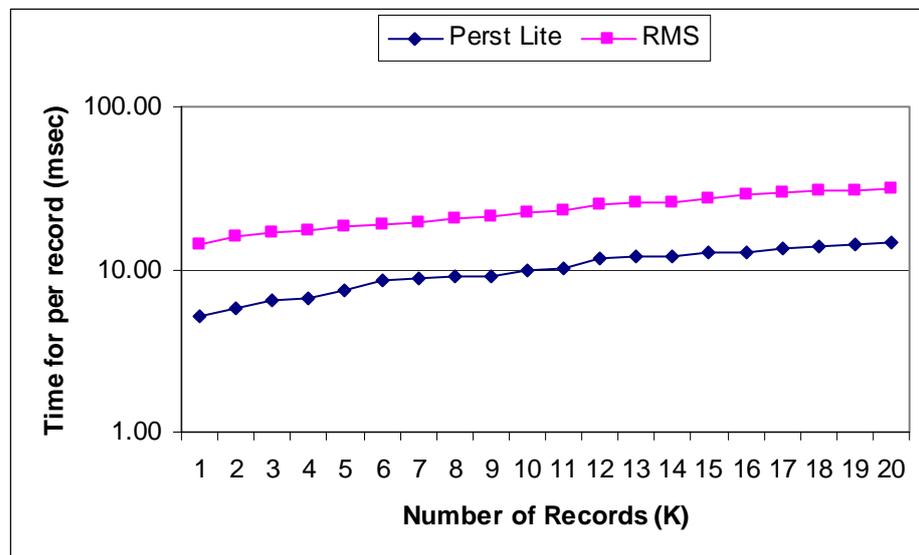


Figure 20: The Insert Function Comparison of Perst Lite and RMS on Mobile

5.2.2.1 Insert Function

Similar with the insert function evaluation in the simulation, the execution time in the mobile is only measured as the exact insert time for each insert operation, rather than for the whole application. The outcome is the average time for the insert operation of each record at each assessment and its unit of measure is msec.

Figure 20 shows the insert function performance comparison of Perst Lite and RMS on the Nokia N73. It is interesting to note that Perst Lite reflects slightly

better performance than does RMS because of the pool size set at 64 K in Perst Lite and its database engine. It is also observed that the running time for both Perst Lite and RMS rises with the increase in the record number. This is different from the performance in the simulation. The reason might be the application run in the memory of the mobile phone, rather than the external disk.

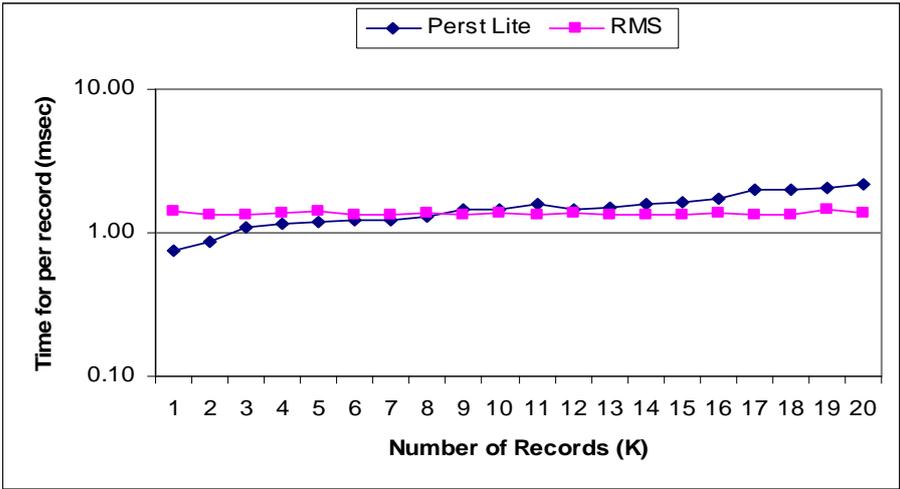


Figure 21: The Random Search Comparison of Perst Lite and RMS on Mobile

5.2.2.2 Sequence Search Function

The functionality of the sequence search in the mobile phone is the same as in the simulation. RMS puts all of the records into the enumerator and searches them from the beginning to the end. Perst Lite sets all of the records into the iterator and retrieves them. The execution time is measured for the search operation’s time.

The outcome is the average time for the sequence search operation of each record at each assessment and its unit of measure is msec.

The sequence search performance of Perst Lite and RMS on the mobile phone is illustrated in Figure 21. It appears that the Perst Lite performance and the RMS performance are quite close. RMS performance remains constant, whereas Perst Lite rises somewhat with increases in the record number. It is obviously different

from the result from the simulation. The reason may be in that the database is stored into the mobile phone memory, whereas in the simulation it is stored in an external disk. This reduces the I/O operation significantly for RMS.

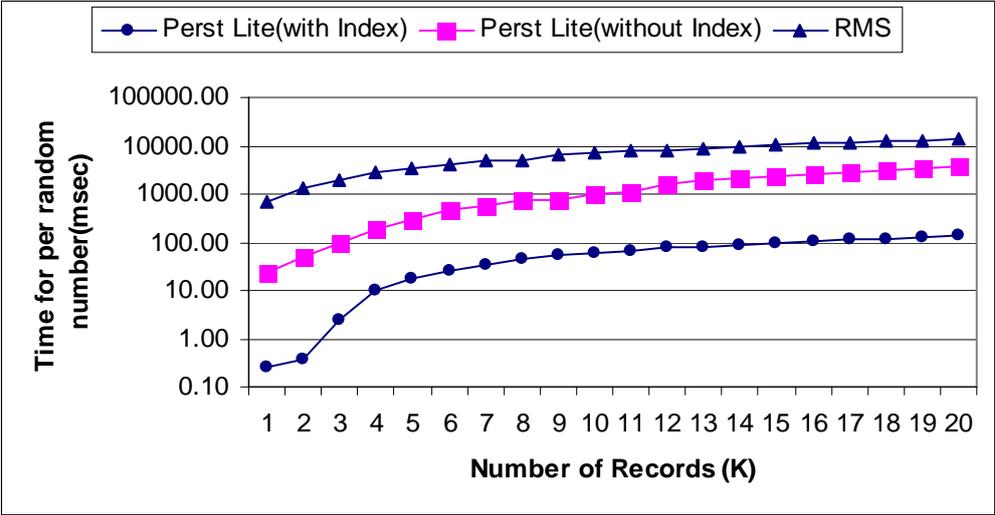


Figure 22: The Random Search Comparison of Perst Lite with Index and without Index and RMS on mobile

5.2.2.3 Random Search Function

The execution time, outcome, and unit of the random search operation are similar to the simulation.

Like the random search function of Perst Lite in the simulation, the real device evaluation is also evaluated as Perst Lite with an index and Perst Lite without an index. The operation is the same as in the simulation. There are 1,000 random numbers generated for the Perst Lite and RMS random search function. The outcome is the average time for 1,000 search times at each assessment.

Figure 22 illustrates the performance comparisons of the random search function of RMS, Perst Lite with an index, and Perst Lite without an index. It is obvious that the random search of Perst Lite with an index increases significantly with the increase of the number of records, particular from 2,000 to 40,000 records. The execution time for 20,000 records is 550 times that for 1,000 records. The random

search function of Perst Lite without an index also increases noticeably with an increase in the number of records. The execution time of the maximum number of records is 150 times that of the minimum. The running time of the RMS random search rises slowly with increases in the number of records compared with Perst Lite with an index and without an index. The execution time for the maximum record number is 20 times that of the minimum.

It is interesting to note that the random search of Perst Lite with an index and without an index show a significantly better performance than does RMS. The database engine of Perst Lite is the major reason for this. Furthermore, the Perst Lite random search without an index takes about 20 times the one with an index. The random search of Perst Lite with an index takes less time, because the index engine helps the user to locate the record quickly.

5.2.2.4 JAR Size Comparison

A Java archive (JAR) file contains many compressed files in the stored file. In the mobile application, this file is created in a test environment and then deployed to the mobile phone.

Table 14 compares the Jar file sizes of Perst Lite and RMS deployed into the mobile phone. It is noted that Perst Lite and RMS have the fixed size of 1339 KB and 1053 KB, respectively. They do not change with an increase of the record number.

Table 14: The Jar File Size Comparison for Perst Lite with Index and without Index and RMS

Record Number	Perst Lite (Deployed size,KB)	RMS (Deployed size, KB)
1000	1339	1053
2000	1339	1053
3000	1339	1053
4000	1339	1053
5000	1339	1053
6000	1339	1053
7000	1339	1053
8000	1339	1053
9000	1339	1053

10000	1339	1053
11000	1339	1053
12000	1339	1053
13000	1339	1053
14000	1339	1053
15000	1339	1053
16000	1339	1053
17000	1339	1053
18000	1339	1053
19000	1339	1053
20000	1339	1053

5.2.3 Comparison on 5,000 Records

Table 15 compares the insert function, sequence, and random search function for Perst Lite and RMS at 5,000 records.

Table 15: The Comparison of Perst Lite and RMS on the Simulation and the Mobile at 5,000 Records

Compared Item		Perst Lite (ms)	RMS (ms)	
Simulation	Insert function	19.9	117.78	
	Sequence search	0.38	32.03	
	Random search	with index	2.25	56222.08
		without index	87.66	
Nokia N73	Insert function	7.49	18.26	
	Sequence search	1.17	1.4	
	Random search	with index	18.12	3305.15
		without index	286.34	

It is observed that Perst Lite performance is much better than RMS performance in terms of the insert function, the sequence search, and the random search with and without an index based on the simulation. This finding is particularly the case in random search. The first explanation is that Perst Lite uses 64K as the pool size. It effectively reduces the load of I/O, which results in a decrease in the access time. Next, Perst Lite has an index engine, which is efficient in locating records quickly.

In contrast, on Nokia N73, the performance of RMS is improved significantly. The performance of the sequence search of Perst Lite is similar to that of RMS. The execution time of the insert function of RMS is only twice that of Perst Lite. The major reason might be that the databases are stored in the mobile phone memory, rather than in the extended memory. It reduces I/O operations when manipulating records. On the Nokia N73, however, the performance difference between Perst Lite and RMS on a random search with and without an index is still significant. The index engine provided by Perst Lite and its database engine are the major reasons for this.

Perhaps if the mobile application is stored into the miniSD card in the mobile phone, the performance difference between Perst Lite and RMS might be similar to that of the simulation, because it needs all I/O operations. It is, however, not common to use an extended memory card in the mobile phone, so this project does not investigate this aspect further.

5.2.4 Database Discussion

In general, Perst Lite shows very good performance for the insert function, the sequence search, and the random search with an index and without an index at each assessment, particularly in the random search.

As mentioned in the system design, the data storage process of the application is run as a background process. Thus, the insert performance can be neglected in the application. The application allows the user to enter a specific date and retrieve the data related to that date from the database. So, the random search performance becomes essential for the choice of the embedded database. As can be seen from the comparisons mentioned above, the Perst Lite random search performance is much faster than that of the RMS, particularly when using the index for the search. Thus, Perst Lite is the most suitable choice for the embedded database in the project.

It is observed that, in the Nokia N73, the performance of the random search with an index of Perst Lite reduces somewhat if the number of records exceeds 17,000, whilst the performance of the random search without an index of Perst Lite decreases noticeably when the number of records reaches 12,000.

On the other hand, it is supposed that the sampling of the sensors is set at the rate of once per hour in the project, so that each day will have 24 records, with two years of data reaching 17,520 records ($24 \times 365 \times 2 = 17520$). Considering the features mentioned above, it is suggested that the maximum record number in the project is limited to 18,000, based on the per hour sampling.

5.3 Project Performance Evaluation

Once the database has been chosen and the size of the data amount is confirmed, the performance of the project becomes the most important factor.

5.3.1 Analysis of the Project

To evaluate the project, a number of issues need to be considered, including the application initialisation, data receiving, data storage, data search, data selection, user interface response, and data plotting.

Firstly, the initialisation of the application is one of elements used to measure performance. The initialisation measures the execution time from the beginning of the application to reaching the welcome menu. It contains parameter initialisation and the application activation.

Next, the data receiving and data storing process are other attributes used to measure the project performance. Nonetheless, the data process in the application is set to run as the background, so the user does not face the performance issue directly. Thus, the performance of the data receiving and storing is neglected in the project.

The performance of the data search from the database is one of the most important issues in the application. The execution time for the search of the specific date is measured from the embedded database entered by the user. It is one crucial performance of the application.

The performance of the data selection is used for the process of choosing 48 data points from the retrieved data, as mentioned in the previous paragraph. The results are variable and dependent on the searching range.

The user interface response is another essential topic of the project. It is related to whether or not users like to use the application. Through the test, it is interesting to notice that the response time of all interfaces is less than 0.5 seconds. The timing is taken from the time the user presses the key to the next menu on the mobile screen. The user interfaces related to the application are designed by J2ME Polish, except for the data plotting and the customising user interface. Thus, it is not necessary to measure the performance of user interfaces, except for the data plotting. Obviously, the performance does not change with the data range.

The last important performance to be considered is that of the data plotting. It plots the 48 data points retrieved from the embedded database on the small mobile screen. The two types of graphics; line chart and bar chart; are customised for the project. Thus, the performance is worthy of investigation.

The evaluation was supposed to be implemented in the simulation and in the real device, a Nokia N73. It is, however, interesting to notice that the evaluation based on the simulation crashed when the record number exceeded 5,000. The limitation of defaultColorPhone in the simulation might be the reason for this. Thus, the evaluations were only run on the Nokia N73.

The record number for the experiments varies from 1,000 to 18,000 and the evaluation was implemented at each thousand records. The retrieved date range includes one day, two days, five days, ten days, fifteen days, twenty days, twenty-five days, and thirty days. The experiment on 1,000 records was not evaluated at the 20, 25 and 30 day points due to the limited number of records. The unit of measurement of all the evaluations is msec.

There were 18,000 records generated for the experiments, with the first data point being from midnight of 1st January, 2007. Each record's format was generated according to the database format mentioned in Section 4.2.1. The time stamp attribute was set as the index. Every date contains 24 data points (sampling is per hour, as mentioned in Chapter 3). Thus, the retrieved data increases with any increase in the retrieved range.

For the evaluation of the data plotting, 100 random dates were generated and the values were limited to the period from 1st January, 2007 to the last date of the random search mentioned in Table 16, according to the various numbers of records.

Table 16 shows the maximum of the random retrieved date for the data search. The duration for the random search is the bound of the random date generated for retrieval.

Table 16: The Maximum of the Random Retrieved Date for Data Search

Numbers of record	Days	Months	Last Date for the random search
1000	41.6	1.3	2007. 01. 20
2000	83.3	2.8	2007. 02. 28
3000	125	4.2	2007. 03. 28
4000	166.6	5.5	2007. 04. 28
5000	208.3	6.9	2007. 06. 28
6000	250	8.2	2007. 07. 28
7000	291.6	9.6	2007. 08. 28
8000	333.3	11	2007. 10. 28
9000	375	12	2007. 11. 28
10000	416.6	13.7	2007.12.28
11000	458.3	15.1	2008.02.28
12000	500	16.5	2008.03.28
13000	541.6	17.8	2008.05.28
14000	583.3	19.2	2008.06.28
15000	625	20.5	2008.07.28
16000	666.6	21.92	2008.09.28
17000	708.3	23.3	2008.10.28
18000	750	24.6	2008.11.28

5.3.2 Initialisation Performance

The initialisation is the first step to the application, and the first measurement for the application performance. The initialisation is measured from the application start to reaching the welcome menu. It includes the initialisation of the parameters and the activation of the application.

Figure 23 compares the initialisation performance of MicroBaseJ. It is observed that the execution times of the initialisation are approximate. It is interesting to note that the performance of the initialisation is quite stable, and that it does not change with the

increase in the record number, or the increase in the search duration. It is also observed that the execution times of the initialisation are quite long (between 4.4 and 4.8 seconds), which exceeds the user requirement for a response within 2 seconds. The poor performance of the Nokia N73 in all applications is one of the major reasons for this slow speed. Another is that there are a number of parameters to be initialised. The last reason might be due to the poor performance of JAVA, however, the slow initialisation can be tolerated by users, because they are very sensitive to the interface response rather than to the initialisation.

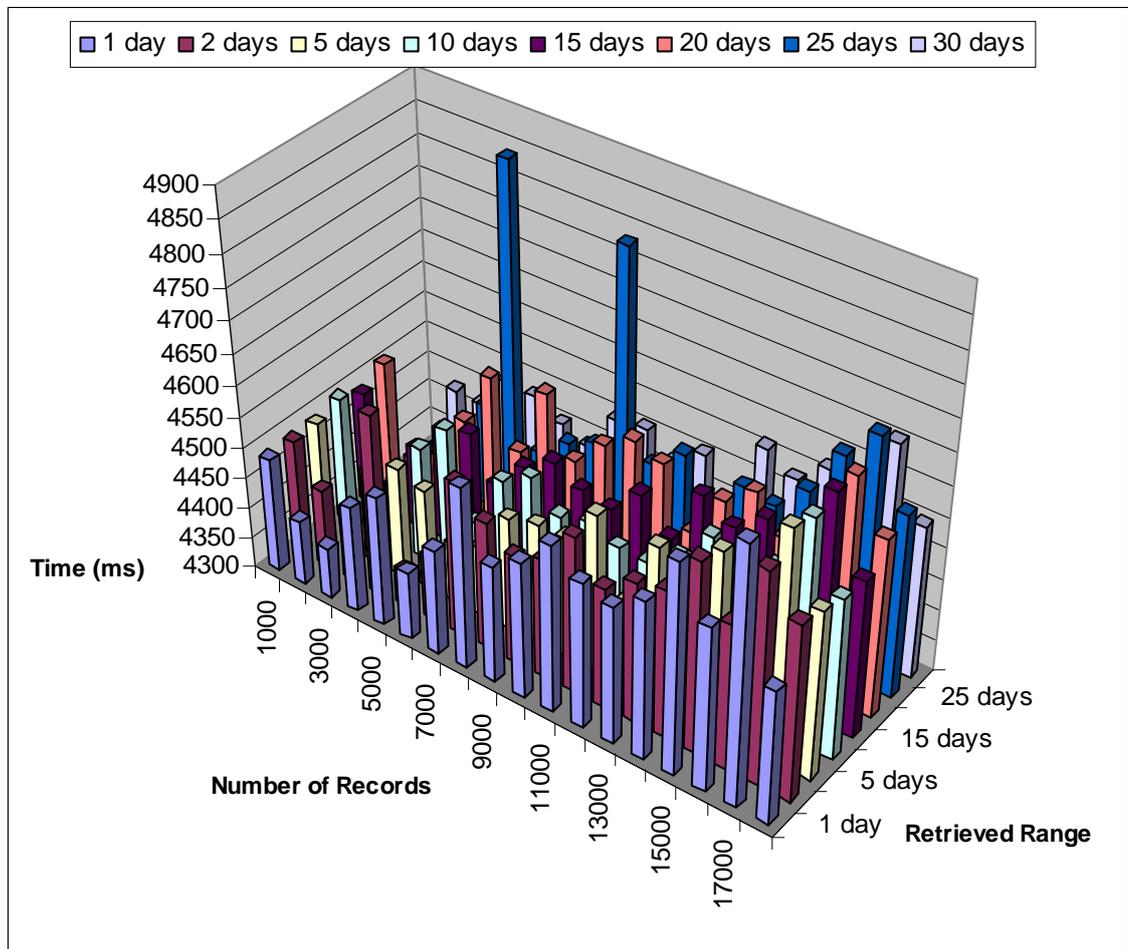


Figure 23: The Initialisation Performance Comparison

5.3.3 Data Search Performance

The test here generated 100 random dates at the various date range and the various record number. The range of the 100 random dates was limited to the duration mentioned in Table 16. The performance of the data search is measured by the whole process of the data search from the embedded database in terms of the specific date.

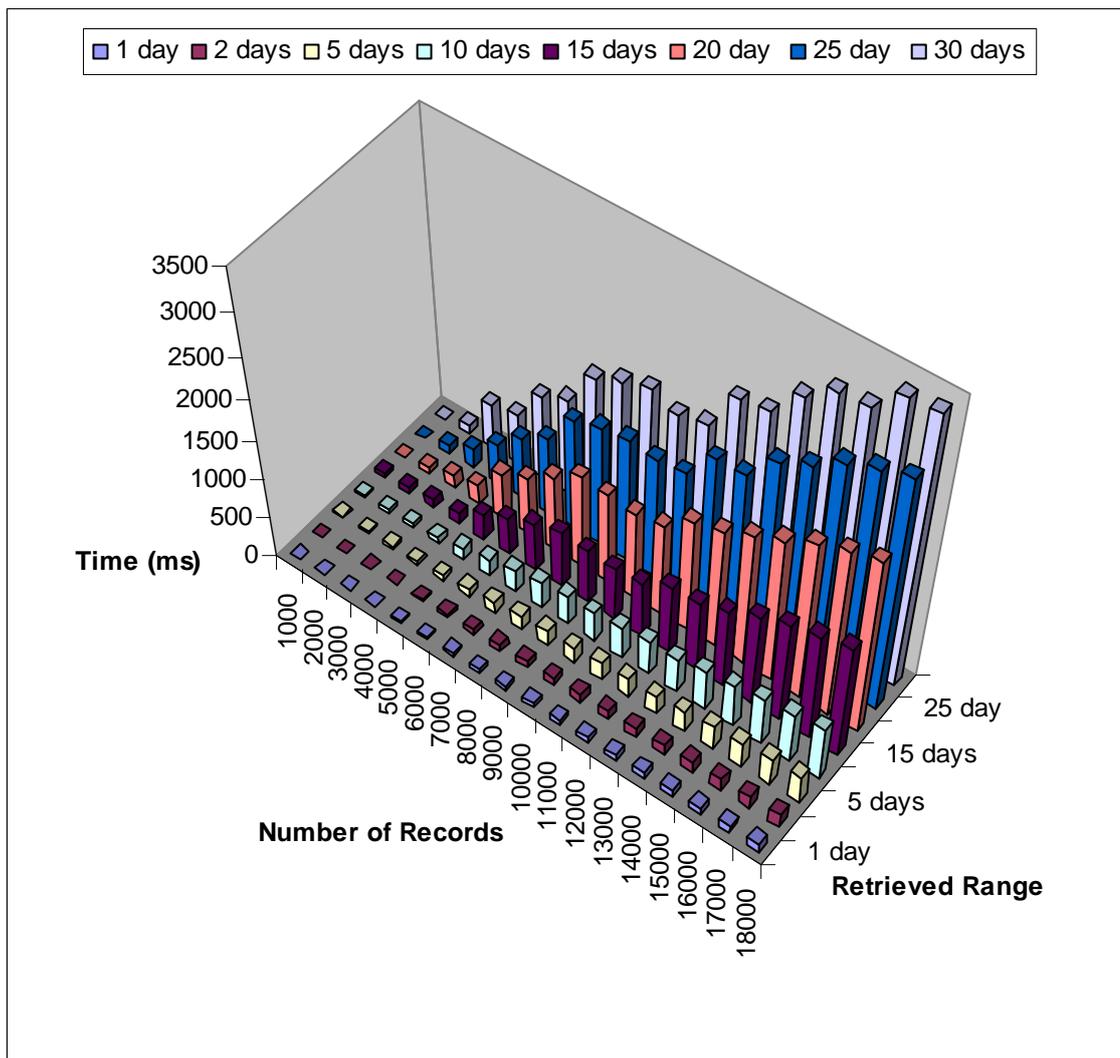


Figure 24: The Data Search Performance Comparison

Figure 24 compares the performance of the data search for MicroBaseJ. It is interesting to note that the execution time of the data retrieval rises slightly with the increase of the record number, while the execution time for the data search goes up significantly with

the increase of the retrieved range. It is observed that the execution time of the retrieved data in the 18,000 records is around 20 times that of the 1,000 records in the various date ranges. In general, the performance of the data retrieval shows that the execution times are all less than 2 seconds when the retrieved range is limited to 20 days in the various record numbers, or where the record number is under 12,000 in the various retrieved ranges.

5.3.4 Data Selection Performance

The evaluation measured the performance of the 48 data points' selection from the outcome of the embedded database. There are 24 data points per day in the database, and the amount of the retrieved data from the database varies in terms of the various retrieved ranges. The one day range has 24 data points, while the 30 day range has 720 points, however, the mobile phone can only present 48 data points at a time. Thus, it is necessary to select only 48 points from the retrieved outcomes in the database.

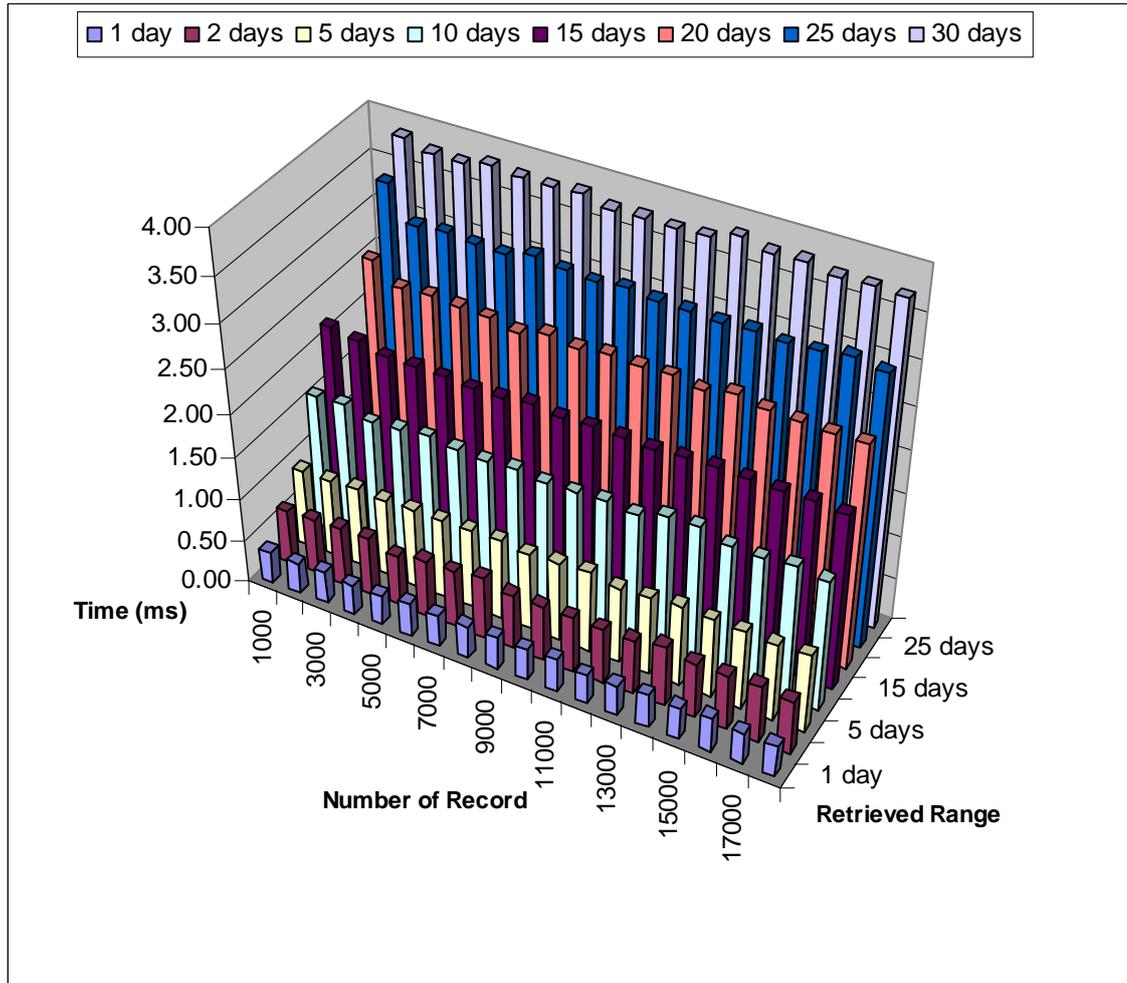


Figure 25: The Data Selection Performance

Figure 25 compares the performance of the data selection. It is interesting to note that the data selection's performance reduces with the increase of the retrieved range, whereas the increase of the record number does not affect the performance of data selection in the same retrieval range. The explanation accords with the application's design principle, where the outcome record numbers are the same at the same retrieval range. It is observed that the performance of the data selection is excellent, with the maximum value being less than 0.04 seconds. Thus, the performance can be neglected in the application.

5.3.5 Data Plotting

The data plotting is another key element contributing to the performance of the application. It is a customised widget for the graphics of the 48 data points. It measures

the execution time for the whole process of plotting the 48 data points on the small mobile screen. The data plot includes the line chart and the bar chart.

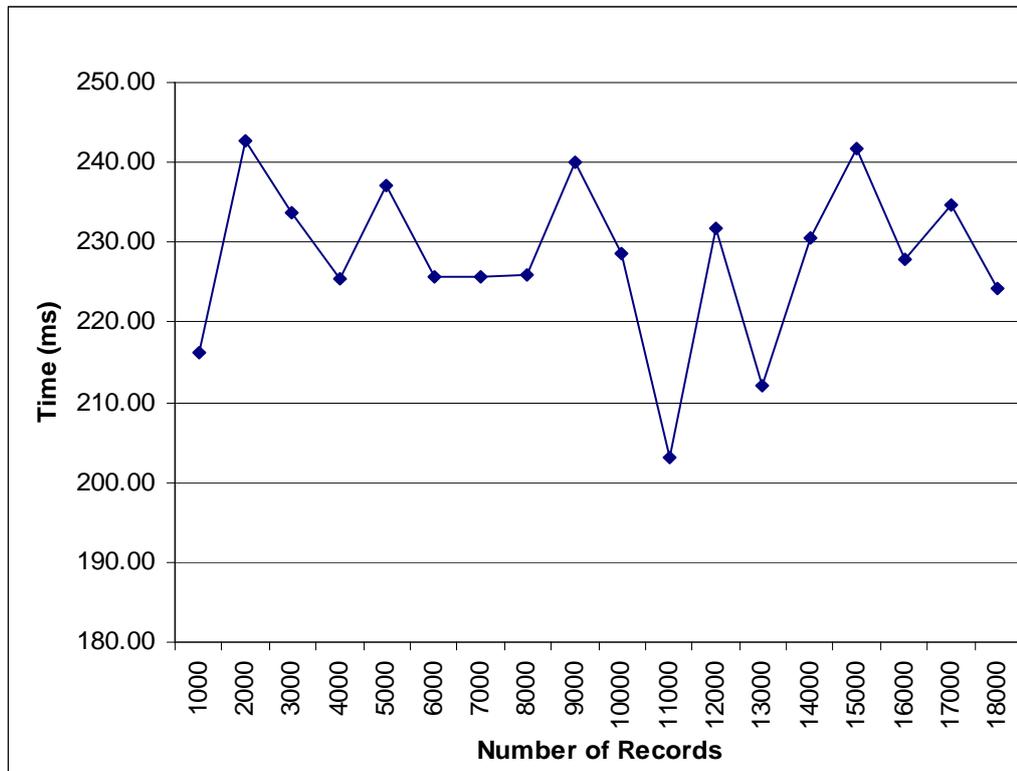


Figure 26: The Data Plotting Performance Comparison

Figure 26 compares the performance of the data plotting for MicroBaseJ. As there are only 48 data points selected for the graphics on the various retrieved date ranges, using a 2D figure to present the relation of timing with the record number is adequate. It is observed that the execution times of the data plotting fluctuate around 0.2 to 0.25 seconds with the increase of the record number. The fluctuation may be due to the mobile phone running condition.

5.3.6 Discussion of the Project

Overall, the performance of the project satisfies the user requirement for response within 2 seconds. The majority of the user interface responses are less than 2 seconds, with the exception of the application initialisation and a large retrieval range, such as more than 20 days. The poor performance of the initialisation is caused by a number of factors, such as the mobile phone performance and features, the feature and

performance of the development language, and the number of parameters to be initialised. Usually, the slow performance of the initialisation can be accepted by users. It is proposed that the retrieved data range is limited to up to 20 days to satisfy the user requirement for a response within 2 seconds.

6 Conclusion

6.1 Introduction

The objective of the thesis is to develop a remote monitoring system based on the mobile phone. This should integrate embedded database and friendly graphic user interfaces for the horticulture application. In this chapter the main outcomes of the research are summarised. This is followed by the conclusion, which is based on the results obtained. It also includes recommendations for further work.

6.2 Conclusion

MicroBaseJ is a remote monitoring system based on the mobile phone. This thesis proposed the infrastructure of MicroBaseJ. First, it analysed the requirement for MicroBaseJ. User requirements were collected and analysed. After that, the general architecture was proposed. The data sources contain ten items, including four temperatures, humidity, soil moisture, wind direction and speed, rainfall, and leaf wetness. The communication protocol between the Sentinel and the mobile phone was proposed to use SMS, with Bluetooth suggested for the communications between the mobile phone and the backup server. In addition, functional requirements and non-functional requirements were analysed. The interface description, the transferred data format between components, and the dataflow model for the project were also discussed. The system was designed in terms of UML, and the class diagram and state diagram were proposed and discussed.

The details of the embedded database, including the data format and the data storage, were discussed. MicroBaseJ used Perst Lite, the object-oriented database, as the database embedded into the mobile phone. There are eleven attributes, including a time stamp plus ten items from sensors, in this application. The time stamp was set as the String format and the retrieved index. The other attributes' formats are set as the Integer type. The hardware and software for user interfaces were investigated. This thesis also proposed a minimum resolution for the mobile phone based on MIDP 2. The general

ideas for the screen layout, the command and the menu of user interfaces were analysed and discussed based on this. The structure and design idea for one example of the graphic user interfaces were also analysed and discussed, including the position and size of each item in the screen. Overall, these graphic user interfaces are flexible, portable, scalable, and controllable.

The performance of two embedded databases, Perst Lite and RMS, were compared based on the simulation and the Nokia N73. The evaluations included the insert function, the sequence search function, and the random search function. The record number for these tests varies from 1,000 to 20,000, with evaluations implemented at each thousand records. Their storage sizes were also compared. Overall, Perst Lite shows a very good performance in the insert function, the sequence search function, and the random search with an index and without an index at each assessment, especially in the random search function either on the simulation, or the Nokia N73. The comparisons show that the Perst Lite random search performance is faster than the RMS, particularly with an index. Thus, Perst Lite is the most suitable choice for the embedded database in this project. According to the performance comparisons, it is suggested that the maximum record number be 18,000 for MicroBaseJ. This allows a capacity of two years data at a point of one sample per hour.

Finally, the project performance was investigated. The evaluation for this project is focused on the mobile side. Its performance was evaluated according to the application initialisation, the data search, the data selection, and the data plotting. These evaluations were implemented on the Nokia N73. The record number for experiments varies from 1,000 to 18,000 and the evaluation is implemented at each thousand records. The retrieved date ranges include one day, two days, five days, ten days, fifteen days, twenty days, twenty-five days, and thirty days. There are 100 random dates created for the data search evaluation.

For the initialisation, the performance is quite stable. Their values range from 4.4 to 4.8. For the data search, the execution times are all less than 2 seconds when the retrieved range is limited to 20 days in the various record numbers, or the record number is under 12,000 records in the various retrieved ranges. The execution times of the data selection are less than 0.04 seconds. The running time of the data plotting fluctuates around 0.2 to 0.25 seconds. The results show that the response times of most user interfaces are less

than 2 seconds, with the exception of the application initialisation and the large retrieval range exceeding 20 days. It seems that this satisfies the user requirement for a response with 2 seconds. It was proposed that the retrieved date range was limited to up to 20 days.

6.3 Future Work

The embedded database is an important issue of this project. The object-oriented database, Perst Lite, was evaluated and compared with RMS, based on the insert function, the sequence search function, and the random search function run on the simulation and the Nokia N73. There are several embedded databases. For example, the SQL embedded database, Pointbase, need to be investigated and evaluated.

Another key topic is the graphic user interface. Only the line chart and the bar chart are investigated in this application. The pie chart needs to be explored as well. There is an additional issue of allowing the graphics to be more flexible, portable, and scalable, or even to generate the customising library for the graphics.

The information collection element is based on the TINI and it is wired. How to migrate it to a wireless sensor network is another important area of research for the remote control mobile application.

References

- [1] Ravi S, Chathish MS, Prasanna H. WAP and SMS based emerging techniques for remote monitoring and control of a process plant. ICSP '04; 2004 August 31-September 4; Beijing, China; 2004. p. 2672-5.
- [2] Bass FM. A new product growth model for consumer durables. *Management Science* 1969;15:215-27.
- [3] Nikolova M, Meijs F, Voorwinden P. Remote mobile control of home appliances. *Consumer Electronics*. 2003;49(1):123-7.
- [4] Nichols J, Myers B. Controlling Home and Office Appliances with Smart Phones. *PERVASIVE computing*. 2006;5(3):60-7.
- [5] Ishikawa S, Saito Y, Cohen M. Mixed-reality "party-line night club" - synchronization of networked avatars and appliances with mobile phone ringtones: integrating Java3D and LAN-tap roomware with J2ME. In: Saito Y, editor. *CIT 2005* May 23-26; Binghamton University; 2005. p. 553-7.
- [6] Lin CC, Chen MS. On controlling digital TV set-top-box by mobile devices via IP network. In: Ming-Syan C, editor. *Multimedia, Seventh IEEE International Symposium*; 2005 December 12-14; Irvine, CA, USA; 2005. p. 8.
- [7] Sriskanthan N, Meher PK, Ng GS, Heng CKAHCK. WAP-teletext system. *Consumer Electronics, IEEE Transactions* 2004;50(1):130-8.
- [8] Pires JN. Remote monitoring and inspection of robotic manufacturing cells. *ASME'01*; 2001 July 8-12; Como, Italy; 2001.
- [9] Carswell JD, Eustace A, Gardiner K, Kilfeather E, Neumann M. An environment for mobile context-based hypermedia retrieval. In: Eustace A, editor. *Database and Expert Systems Applications 2002* September 2-6; France; 2002. p. 532-6.
- [10] Rahman SA, Bhalla S. Supporting spatial data queries for mobile services. In: Bhalla S, editor. *IEEE/WIC*; 2005 September 19-22; France; 2005. p. 696-9.
- [11] Hung K, Zhang Y. Implementation of a WAP-Based telemedicine system for patient monitoring. *Information Technology in Biomedicine*. 2003 June;7(2):101-7.
- [12] Andrade R, von Wangenheim A, Bortoluzzi MK, De Biasi HH. A strategy for a wireless patient record and image data. *International Congress Series*. 2003;1256:869-72.

- [13] Koop A, Mosges R. The use of handheld computers in clinical trials. *Controlled Clinical Trials*. 2002;23(5):469-80.
- [14] Kogure Y, Matsuoka H, Kinouchi Y, Akutagawa M. The development of a remote patient monitoring system using Java-enabled mobile phones. *Engineering in Medicine and Biology 27th Annual Conference*; 2005 September 1-4; Shanghai, China; 2005. p. 2157-60.
- [15] Lin C, Chiu M, Hsiao C, Lee R, Tsai Y. Wireless health care service system for elderly with dementia. *Information Technology in Biomedicine*. 2006 October 10(4):696-704.
- [16] Ketamo H. xTask-adaptable working environment. *Wireless and Mobile Technologies in Education*; 2002 August 29-30; Sweden; 2002. p. 55-62.
- [17] Black JT, Hawkes LW. A prototype interface for collaborative mobile learning. *IWCMC'06*; 2006 July 3-6; Vancouver, British Columbia, Canada; 2006. p. 1277-82.
- [18] Bollen L, Eimler S, Ulrich Hoppe H. SMS-based discussions - technology enhanced collaboration for a literature course. In: Eimler S, editor. *Wireless and Mobile Technologies in Education*; 2004 March 23-25; Taiwan; 2004. p. 209-10.
- [19] Meurant RC. Cell phones in the L2 classroom: thumbs up to SMS. *International Conference on Hybrid Information Technology*; 2006 November 10-11; Korea; 2006.
- [20] Gao BK, Wang XF, Jiang JG, SUN ZQ. Development of the mobile Web service distribution system platform-independence. *Machine Learning and Cybernetics*; 2004 August 26-29 Shanghai, China; 2004. p. 7-9.
- [21] Bakos B, Farkas L, Nurminen JK. Search engine for phonebook-based smart phone networks. In: Farkas L, editor. *Vehicular Technology Conference 2005* May 30 - June 1; Stockholm, Sweden; 2005. p. 2795-9 Vol. 5.
- [22] Rahman SA, Bhalla S, Hashimoto T. Query-by-object interface for dynamic access and information requirement elicitation. In: Bhalla S, editor. *ICMB 2005* July 11-13; Sydney, Australia; 2005. p. 667-70.
- [23] Lo CTD, Chang M, Frieder O, Grossman D. The object behavior of Java object-oriented database management systems. In: Chang M, editor. *Information Technology: Coding and Computing 2002* April 8-10; Nevada, USA; 2002. p. 247-52.
- [24] Imai Y, Ooga M, Yamane D, Sadayuki O, Iwamoto Y, Masuda S. Mobile phone-enhanced user interface of remote monitoring system. In: Ooga M, editor. *ICMB 2005* July 11-13; Sydney, Australia; 2005. p. 63-8.
- [25] Yang C, Kou M. Remote Monitoring and Control of PC Clusters Using Mobile Phones with J2ME. *ITRE*; 2005 June 27-30; Taiwan; 2005.
- [26] Rahman SA, Bhalla S, Hashimoto T. Query-by-object interface for information requirement elicitation in m-commerce. *CEC*; 2005 July 19-22; Germany; 2005. p. 143-50.

- [27] Howard T, Bradford PG. PUC to J2ME interface generator. SoutheastCon; 2007 March 22-25; VA, USA; 2007. p. 116-20.
- [28] Mayora-Ibarra O, Paz-Arroyo Odl, Edgar Cambranes-Martinez, Fuentes-Penna A. A visual programming environment for device independent generation of user interfaces. *CLHC*. Rio de Janeiro, Brazil: ACM 2003.
- [29] Bollen L, Eimler S, Hoppe HU. The use of mobile computing to support SMS dialogues and classroom discussions in a literature course. ICALT; 2004 August 30 - September 1; Joensuu, Finland; 2004. p. 550-4.
- [30] Bluetooth SIG. Basics. N. D. [cited 2007 May 12, 2007]; Available from: <http://www.bluetooth.com/Bluetooth/Technology/Basics.htm>
- [31] Alhakim MM, Al-Kittani I, Bakleh A, Swidan M, Zarka N. Bluetooth remote control. ICTTA; 2006 April 24-28 Syria; 2006. p. 2674-7.
- [32] Tan K, Soh CY. Internet home control system using Bluetooth over WAP. *Engineering Science and Education*. 2002 August 2002;11(4):126-32.
- [33] Open Mobile Alliance. WAP White Paper. 2000 [cited 2007 May 18]; Available from: http://www.wapforum.org/what/WAP_white_pages.pdf
- [34] Nikolakopoulos G, koudourakis M, Tzes A. An integrated system based on WEB and/or WAP framework for remote monitoring and control of industrial processes. *International Symposium on Virtual Environments, Human-Computer Interfaces, and Measurement Systems*; 2003 27-29 July; Lugano, Switzerland; 2003.
- [35] Sun Microsystems. Java SE at a glance. N. D. [cited 2007 February 12]; Available from: <http://java.sun.com/javase/>
- [36] Zan J. Java ME core technology and best practices. Beijing: Publishing house of Electronics Industry 2007.
- [37] Sun Microsystems. Java ME Platform Overview. N. D. [cited 2007 April 2]; Available from: <http://java.sun.com/javame/technology/index.jsp>
- [38] Qualcomm Incorporated. BREW: bring wireless services to life. 2008 [cited 2008 May 20]; Available from: http://brew.qualcomm.com/brew_bnry/pdf/BREW_brochure.pdf
- [39] Ziff Davis Enterprise Holding Inc. Microsoft opens full Windows CE kernel source. 2006 [cited 2007 March 20]; Available from: <http://www.linuxdevices.com/news/NS6932977445.html>
- [40] McObject. McObject releases Perst Lite, an open source embedded database for intelligent devices on Java's J2ME platform. 2006 [cited 2007 March 15]; Available from: <http://www.mcobject.com/pressroom.php?step=3&article=75>
- [41] DataMirror. POINTBASE MICRO: the ultra-compact database for enterprise mobility. 2004 [cited 2007 March 11]; Available from: <http://www.pointbase.com/resourcecenter/pdfs/micro.pdf>

- [42] db4objects. db4o open source object database. N. D. [cited 2007 March 12]; Available from: <http://www.db4o.com/about/productinformation/db4o%20Product%20Information%20V6.0.pdf>
- [43] McObject. Perst Documentation. 2006 [cited 2007 March 18]; Available from: <http://www.mcobject.com/downloads.php?step=6&id=51>
- [44] Knudsen J. Wireless Java: developing with J2ME. Berkeley, CA Apress 2003.
- [45] Stone D, Jarrett C, Woodroffe M, Minocha S. User interface design and evaluation. Amsterdam: Elsevier : Morgan Kaufmann 2005.
- [46] Olsson G, Piani G. Computer systems for automation and control. Hertz: Prentice Hall International (UK) Ltd. 1992.
- [47] Piani G. Computer-supported complexity reduction in process control: a cognitive approach to user interface design. Sweden: Department of Industrial Electrical Engineering and Automation, Lund Institute Technology 2003.
- [48] Miller GA. The magical number seven, plus or minus two - some limits on our capacity for processing information. The Psychological Review. 1956;63(2):81-97.
- [49] Marcus A, Smilonich N, Thompson L. The cross-GUI handbook: for multiplatform user interface design. New York: Addison-Wesley Publishing Company 1995.
- [50] Maxim Integrated Products. Getting started with TINI. 2004 [cited 2007 May 12]; Available from: http://www.maxim-ic.com/products/tini/pdfs/TINI_GUIDE.pdf
- [51] MOBITEK System Sdn. Bhd. GSM modem - Wavecom Fastrack M1306B. N. D. [cited 2007 May 1]; Available from: <http://www.mobitek-system.com/Wavecom/Fastrack.html>
- [52] D'Anjou J, Fairbrother S, Kehn D, Kellerman J, McCarthy P. Java developer's guide to Eclipse. 2nd ed. Boston: Addison-Wesley 2004.
- [53] McAffer J, Lemieux J-M. Eclipse rich client platform: designing, coding, and packaging Java applications. Upper Saddle River, NJ: Addison-Wesley 2006.
- [54] Sun Microsystems. Sun opens Java. N. D. [cited 2007 April 1]; Available from: <http://www.sun.com/2006-1113/feature/story.jsp>
- [55] Sun Microsystems. The Java Programming Language. N. D. [cited 2007 March 25]; Available from: <http://java.sun.com/docs/books/tutorial/getStarted/intro/definition.html>
- [56] Sun Microsystems. Java ME at a glance. N. D. [cited 2007 March 23]; Available from: <http://java.sun.com/javame/index.jsp>
- [57] Liu B. Master Java ME: use Eclipse for mobile development. Beijing: Publishing House of Electronics Industry 2007.

- [58] Virkus R. Pro J2ME Polish: open source wireless Java tools suite. NY: Apress 2005.
- [59] Sommerville I. Software engineering. 8th ed. New York: Addison Wesley 2007.
- [60] National Snow and Ice Data Center. Temperature. N. D. [cited 2007 March 18]; Available from: <http://nsidc.org/arcticmet/glossary/temperature.html>
- [61] National Snow and Ice Data Center. Relative humidity. N. D. [cited 2007 March 21]; Available from: http://nsidc.org/arcticmet/glossary/relative_humidity.html
- [62] Answers Corporation. Soil moisture. N. D. [cited 2007 March 22]; Available from: <http://www.answers.com/topic/soil-water?cat=technology>
- [63] Answers Corporation. Wind speed. N. D. [cited 2007 March 23]; Available from: <http://www.answers.com/topic/wind-speed?cat=technology>
- [64] Answers Corporation. Wind direction. N. D. [cited 2007 March 24]; Available from: <http://www.answers.com/wind+direction?cat=technology>
- [65] Answers Corporation. Rainfall. N. D. [cited 2007 March 20]; Available from: <http://www.answers.com/rainfall?cat=technology>
- [66] New York State Agricultural Experiment Station. Surface wetness duration. N. D. [cited 2007 April 3]; Available from: <http://www.nysaes.cornell.edu/pp/faculty/seem/magarey/leafwet/definition.html#swlw>
- [67] Sommerville I. Software engineering. 6th ed. Harlow, England; New York: Addison-Wesley 2001.
- [68] Object Management Group. Introduction to OMG's unified modeling language. 2007 [cited 2007 November 11]; Available from: <http://www.uml.org/>

Appendix I – Sample Mobile Phones and their Resolutions

Model	Technology	Screen Size
Motorola A1000	GSM	208x320
Motorola A760		320x240
Motorola C380	GSM/GPRS	128x128
Motorola C650	GSM/GPRS	128x128
Motorola E398	GSM/GPRS	176x220
Motorola i335		128x128
Motorola i355		128x128
Nokia 3152	AMPS, CDMA	128x160
Nokia 3155i	AMPS, CDMA	128x160
Nokia 3220	GSM	128x128
Nokia 3230	GSM	176x208
Nokia 3250	GSM/GPRS/EDGE	176x208
Nokia 6560	AMPS, TDMA	128x128
Nokia 6600	GSM	176x208
Nokia 6651	GSM	128x160
Nokia 6670	GSM	176x208
Nokia 6822	GSM	128x128
Nokia E50	E-GSM	240x320
Nokia E60	GSM	352x416
Nokia E61	GSM	320x240
Nokia E62	GSM	320x240
Nokia E70	GSM	352x416
Nokia N70	GSM	176x208
Nokia N71	GSM	320x240
Nokia N72		176x208
Nokia N73		240x320
Nokia N76	GSM/GPRS/EDGE, W-CDMA	128x160
Nokia N80	GSM, W-CDMA	352x416
Nokia N91	GSM, W-CDMA	176x208
Nokia N93	GSM/GPRS/EDGE, W-CDMA	240x320
Nokia N95	GSM, W-CDMA	320x240
Siemens S65	GSM/GPRS	132x176
Siemens S66	GSM/GPRS	132x176
Siemens S6C	GSM/GPRS	132x176
Sony Ericsson K500c	GSM/GPRS	128x160
Sony Ericsson K510	GSM/GPRS	128x160
Sony Ericsson K550	GSM/GPRS/EDGE	176x220
Sony Ericsson K600	GSM/GPRS	176x220
Sony Ericsson K660i	GSM/GPRS, UMTS/HSDPA	240x320

Sony Ericsson K790	GSM/GPRS/EDGE	320x420
Sony Ericsson K800	GSM/GPRS	320x420
Sony Ericsson P900	GSM/GPRS	208x320
Sony Ericsson P908	GSM/GPRS	208x320
Sony Ericsson W300	GSM/GPRS/EDGE	128x160

Appendix II – The Introduction of the User Interfaces

User interface design is one of the most important research works for MicroBaseJ. User interfaces in MicroBaseJ are designed as a cascade. There are a number of user interfaces in MicroBaseJ, including a splash welcome form, the main menu, the items chosen menu, the date selected menu, the data plot menu, the alarm, and the sample rate configuration menu. A splash form is shown on the mobile screen when the application is activated. Figure 27 is the splash form based on the WTK 2.5.1 default ColorPhone simulation.



Figure 27 - The Splash Interface

This form is designed according to the requirement of the industrial partner, MCS Ltd., in order to promote and catch the customers' eyes. It lasts only five seconds and then the main menu is shown on the mobile screen.

Figure 28 shows the main menu of the application. A menu title is shown on the top of the screen. Several graphic items, with brief text for item choice, are presented on the screen to allow user selection. White is set for the background of the screen. The colour

of the graphic items is set at light blue, allowing the user's perception to be more comfortable.



Figure 28 - The Main Menu

There are four functions in this screen, including Charting, Alarm configuration, Sample Configuration, and Quit. The customer uses the Keys Up, Down, Right, and Left, to navigate among these functions. There are two command buttons provided for the user; Select and Quit; on the screen's bottom right and left sides, respectively. The user chooses the particular graphic item and then presses the Select button, then the application is turned to the next menu. If the user chooses Quit + Select, or presses the Quit button, this application is terminated.



Figure 29 - The Item Choice Menu for Chart

If the user chooses Charting and presses the Select button, an interface for charting is shown on the mobile screen. Figure 29 shows the interface that allows the user to select the two specific items for charting. There are ten items in this screen, including four for temperature, two for humidity, wind speed and direction, rainfall, and leaf wetness. Similar to the main menu, all of the icon backgrounds are set as light blue. The customer uses the key Up, Down, Right, and Left to navigate among these icons.

Here two command buttons are provided for the user; Next and Back; on the screen bottom right and left side, respectively. The user chooses the particular two graphic items and then presses the Next button, then the application is turned to the next menu. If the user presses the Back button, the application goes back to the main menu. When the user chooses less, or more than, two icons, an alert form is given to notify the user. After the user's confirmation, Figure 29 is shown on the screen again.

Whenever two items are chosen and the Next button is pressed, another form is shown on the small screen. Figure 30 is the menu to allow the user to type in the specific date and the period for the search. This menu also provides a choice for illustrating the data, as to whether it is to be set as the average value of the data points when their number exceeds 48.

There are three items in this interface; the entered date, the period, and the average choice. The customer uses the key Up, Down, Right, and Left to navigate among these items.

The date of the most recent message is set as the default on the date item, and it also accepts the user's entries. This application checks the validity of the input date. If there is an error, an alert is presented on the screen. After the confirmation, Figure 30 is shown again.

There are two command buttons provided for the user in this interface; Next and Back; on the screen bottom right and left side, respectively. The user clicks on the Next button, and the application turns to the next menu. If Back is pressed, this application moves back to the previous menu.

After the date and the period are selected, the user presses the Next button and the new interface is shown on the screen. The menu illustrates the maximum and minimum value of two items in the specific date and period.

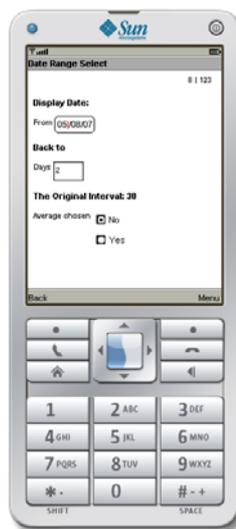


Figure 30 - The Date and Period Entry Menu

There are two command buttons provided for the user in this interface; Next and Back; on the screen bottom right and left side, respectively. The user presses the Next button

and the application moves to the next menu. If Back is pressed this application moves back to the previous menu.

After the user presses the Next button on the maximum and minimum menu, the chart for the two specific items is shown on the screen.

Figure 31 shows three types of chart for this application. Each screen presents two figures. There are two types of chart; the line chart and the bar chart. Any two figures can be combined to form a chart screen.

The line chart is designed for temperature, humidity, soil moisture, wind speed, and direction. Rainfall and leaf wetness are illustrated by the bar chart.

This chart allows the user to interact with the interface using a yellow traceball. The traceball is controlled by the keys on the mobile phone panel. Key 2 and Key 8 are used for the movement of the up and down figures, while Key 4 and Key 6 are used for the navigation of data points to the left or to the right.

When the user controls the trace ball through the key, the text information for that data point' value, including date, time and value, are shown on the top middle of the screen. The colour for the normal data is set as black, whereas the alarm data is presented as red.

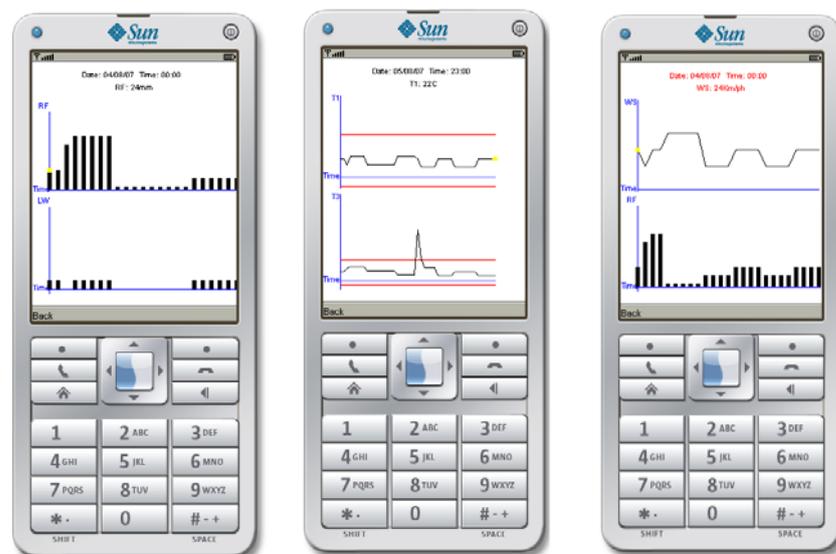


Figure 31 - The Three types of Chart

This interface provides only one button, the Back button. The user clicks it, and the application moves back to the data and period entry menu.

When the user selects the alarm configuration option in the main menu, the item choice menu for the alarm configuration is presented on the screen.

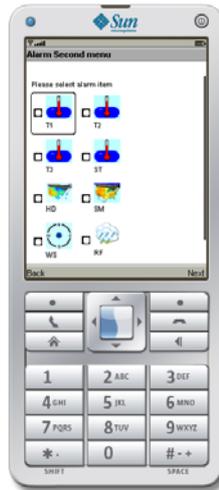


Figure 32 – The Alarm Item Choice Menu

Figure 32 is the interface to allow the user to select the specific alarm item to reset. There are 8 items in this screen, including four for temperature, two for humidity, wind speed, and rainfall. Similar to the main menu, all of the icon backgrounds are set as light blue. The customer uses the key Up, Down, Right, and Left to navigate among these icons.

There are two command buttons; Next and Back; on the screen bottom right and left side, respectively. The user chooses the graphic item and then presses the Next button. The application is then turned to the alarm configuration menu. If the Back button is pressed the application moves back to the main menu.

Figure 33 shows the alarm configuration form. The updated alarm menu shows the current high alarm and the low alarm's value for the specific item, and also allows the

user to change them. The customer uses the key Up, Down, Right, and Left to navigate among the items.

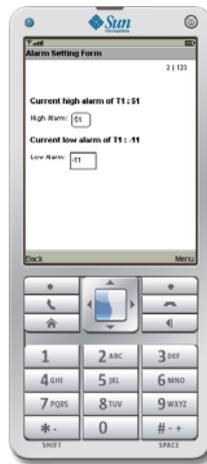


Figure 33 - The Modified Alarm Menu

This interface has two buttons; Next and Back; on the screen bottom right and left side, respectively. The user clicks the Next button and the application is turned to the alarm updated confirmation menu. If the Back button is pressed this application moves back to the alarm item choice menu.

After the user configures the alarm and presses the Next button, Figure 34 is shown on the screen.

Figure 34 shows the updated alarm confirmation interface. It has only one button, the Back button. The user clicks the Back button and this application moves back to the alarm item choice menu.



Figure 34 - Confirmation for Updated Alarm

When the user selects the sample rate configuration option in the main menu, the sample rate configuration menu is presented on the screen, as shown in Figure 35.

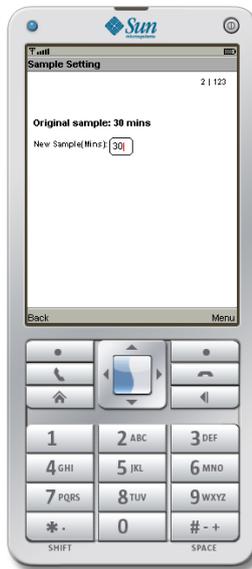


Figure 35 - Sample Rate Modified Menu

The sample rate updated menu shows the current sample rate and allows the user to change to a new one. There are two command buttons; Next and Back; on the screen bottom right and left side, respectively. The user hits the Next button and the application turns to the sample rate updated confirmation menu. If the Back button is pressed this application moves back to the previous menu.

After the user modifies the sample rate and presses the Next button, Figure 36 is shown on the screen.

Figure 36 shows the updated sample rate and the Send button and is provided to confirm that this new sample rate is sent to the Sentinel for the next sampling. After the Send button is pressed the sample rate configuration menu is shown on the screen.

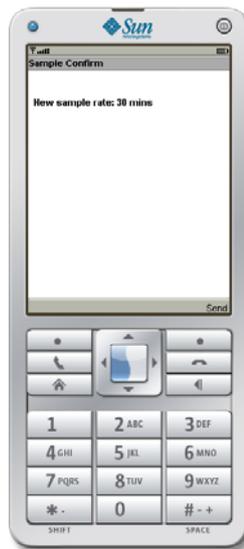


Figure 36 - Confirmation for Sample Rate Updating

Appendix III – The Conference Paper for ATNAC'07

Mobile Embedded Database for Remote Process Management System

Danyi Liu
School of Engineering
AUT University
Auckland NZ
Fmj1496@aut.ac.nz

Adnan Al-Anbuky
School of Engineering
AUT University
Auckland NZ
Adnan.anbuky@aut.ac.nz

Ching Lin
School of Computing and
Mathematical Sciences
AUT University
Auckland NZ
Ching.lin@aut.ac.nz

Abstract—This paper investigates embedded databases for the MicroBaseJ project. The project aims at the development of an integrated database and a user interface for a typical 3G mobile phone with Java MIDP2 capability. This includes methods for data acquisition, mobile data and information communication, data management, and remote user interface. One of the key objectives is to target a generic solution. A number of commercial sectors could benefit from this solution such as horticulture, building management, pollution/water management, industry etc.

Four embedded databases based on J2ME have been investigated. Two of the four have been evaluated and analyzed. The Insert function, Sequence Search and Random Search of Perst List and RMS (Record Management System) databases have been tested. The size of processed data was limited to 20000 records when using the wireless toolkit simulator and 11000 records when using a mobile phone. Perst Lite reflect good performance and has out-performed RMS in all tests.

I. INTRODUCTION

WITH a Global mobile phone user base in excess of 2.8 billion, the current market for information based management processes may be based on at least 75000 applications in the Australasian market in the horticultural and water/irrigation management sectors [1]. Further generic opportunities exist in chilled assets, industry, energy, pollution and security related applications. Mobile applications have covered the area of medicine, education, household appliances control, plant control and spatial information service. However, little has been seen in the horticultural sector.

There has been a number of research work related to the use of cell phone as a remote monitor. Most of them focused on telemedicine, education and plant control. Ravi, Chathish and Prasanna [2] proposed the technical and maintenance personnel supervision and control of machinery and processes from a cellular phone. They used the WAP (wireless access protocol) protocol to develop an alarm management program for providing alert signals when any received data exceeds a preset value for the selected process variables. An implementation of a WAP-based telemedicine system was developed by Hung, Zhang et al.[3]. They

utilized WAP devices as mobile access PCs for common inquiry and patients' common data. Authorized users can view the patient's data, monitor blood pressure and electrocardiogram on WAP equipment in store-and-forward mode. Nikolova, Meijs and Voorwinden [4] developed a technique for interconnecting home and mobile networks to enable the control of home appliances, connected in a home network, from a remote mobile phone or a web pad. The remote control functions include remote mobile programming of VCR, remote mobile control of heating thermostats, remote mobile monitoring using security cameras, etc.

Rahman and Bhalla proposed to use a user-friendly language developed for a Relational Database Management System (RDBMS), Query-By-Example (QBE), to support spatial queries on mobile devices [5]. Lee, Lam and Tsang [6] created a model with a comprehensive wireless distributed real-time database system (WDRDDBS). The model architecture contains a collection of autonomous cells offering database services as in a DRDDBS (distributed real-time database system).

Most applications of mobile phone remote management utilize J2ME (Java 2 Platform Micro Edition) as a tool for user interface implementation. Yang and Kou [7] presented two techniques that uses J2ME to monitor and control PC clusters from mobile phones. Nichols and Myers [8] presented to generate a smart phone interface generator using Microsoft's Windows CE-based Smartphone platform. These interfaces allow users manage each appliance's full functionality and are consistent with other interfaces of the phone.

There are many techniques for communication, including Bluetooth, Zigbee, RFID (Radio frequency Identification), WAP and SMS (Short message service) that relate to the distance between the client and server. Siegemund and Florkeimerier [9] presented an interaction pattern using RFID labels to pass messages. An integrated system based on WEB/WAP framework for remote monitoring and control of industrial processes was proposed by Nikolakopoulos, Koumdourakis and Tzes [10]. User could access to the system using a WEB browser or a WAP-enabled mobile phone.

II. EMBEDDED DATABASE

Databases have provided efficient information retrieval

engines for good number of applications for decades. There are many database applications developed for mainframe, server or even PC. However, there are few applications that have embedded the database into cell phone due to the limitation of resources within the mobile phone. This includes the power, network connection and memory storage.

In recent years, with the development of technology, the price of hardware has dropped significantly. The mobile phones have become more functional and powerful. The embedded database can now be realized in handsets. Currently, there are a few embedded databases available for the handsets, which includes Perst Lite [11], PointBase [12], db4o [13]. RMS (Record Management System) is the basic API of J2ME which can store, retrieve and delete records.

Table 1 provides the comparison of Perst Lite, RMS, PointBase and db4o based on different features. Perst [14] is a simple and object oriented embedded database. It is easy to use and provides high performance. It is intended to be used in applications which need to deal with persistent data in a more sophisticated way than load/store object tree provided by standard serialization mechanism. Perst also provides fault tolerant support (ACID (Atomicity, Consistency, Isolation, Durability) transactions) and concurrent access to the database. Tight integration with

programming language is the main benefit of Perst. Perst stores objects directly without packing/unpacking code (which has to be written for traditional relational databases), so there is no gap between database and application data models. Also Perst (unlike many other OODBMS) does not require a special compiler or pre-processor and provides high level of transparency. Perst Lite is becoming a particular embedded database for mobile phone.

PointBase Micro [12] is a platform-independent Java relational database optimized to run on the Java 2 Micro Edition (J2ME CDC and CLDC/MIDP) and J2SE platforms[12]. It has an ultra-compact footprint (<45K for J2ME MIDP) and can be easily embedded within a Java application, making it transparent to users from the time of deployment. PointBase Micro is ideal for notebooks, PDAs and emerging Java-enabled devices. It provides effective data management for rapid and efficient mobile enterprise applications created by software vendors, systems integrators and corporate IT departments.

db4o [13] is the open source object database that enables Java and .NET developers to slash development time and costs and achieve unprecedented levels of performance. The unique design of db4o's native object database engine makes it the ideal choice to be embedded in equipment and devices,

Table 1 Database Comparison

	Pointbase Micro	Perst Lite	db4o	RMS
Size	45K for J2ME MIDP, 90K for J2ME CDC	30K~300K	250K	Default component of cell phone
Support language	JAVA	JAVA & .net	JAVA & .net	JAVA
platforms	J2EE, J2SE, MIDP, personal JAVA	J2ME, J2SE	JDK 1.1 or later	All J2ME
Application platforms	PDA, mobile handset, PC	PDA, mobile handset	PDA, PC	PDA, mobile handset, PC
Database engine	Relational	Object-oriented	object-oriented database	None
Query language	Subset of SQL92	QBE, perst search method	QBE OR. SOND	None
JDBC	JDBC subset	none	none	none
utilities	Console, others	none	Defragment (not GUI)	none
reflection	yes	none	no	none
Ease of use	good	Very good	Very good	none
flexibility	Cross-platform	Cross-platform	Cross-platform	none
Performance	good	good	poor	Very poor
Open source	none	yes	yes	yes
security	encrypt	encrypt	none	no
strength	Super-small footprint	Easy installation	Easy installation	simple
weakness	SQL and JDBC subset	Non-standard query mechanism	Non-standard query mechanism	Without index, slow search engine
cost	US\$299	US\$2000 for per application per year	US\$100 for personal, US\$ 1000 for corporate	none

in packaged software running on desktop platforms, mobile, and in real-time control systems - in brief, in all Java and .NET environments without database administrator (DBA).

Perst Lite is chosen as the database embedded into handsets. The Perst Lite J2ME database has the simplicity in design and is a high performance database within the resource limit of most intelligent mobile and embedded devices. Perst Lite retains most of the features of the open source Perst since 2003 [14]. These include B-tree, Patricia Trie, Bit index, T-Tree and R-Tree indexes as well as List, Relation, and Set collections, all protected by transactions supporting the ACID properties (Atomicity, Consistency, Isolation and Durability). Perst Lite also offers additional features of multithreaded access, data encryption and asynchronous replication.

III. ARCHITECTURAL DESIGN OF MICROBASEJ

A. MicroBaseJ Architecture

MicroBaseJ project investigates integrated database and user interface for 3G mobile phone with Java MIDP2 capability. It consists of the Cellular Sentinel, mobile phone user interface, server and public wireless network. The architecture of MicroBaseJ is shown in figure 1.



Figure 1 The Architecture of MicroBaseJ.

The Cellular Sentinel collects the information about monitoring and control, and sends these data to the server or mobile phone. The server / mobile phone will receive notification and alarm information from the Cellular Sentinel and store them into database, including temperature, humidity, alarm, GIS (geographic Information System), weather, telemedicine and remote vision, etc. The alert message will present on the mobile phones immediately.

The mobile phone provides a user interface for the subscriber to enquire or update the database, saves the message into local memory and plots the figure locally. Users can access data in the server database through a friendly user interface. The efficiency of different techniques and protocols for mobile phone interface will be compared

and analyzed using metrics of response time, integrity element and cost.

Notifications will be sent to server and saved as backups. And the mobile phone can access the server data through the network.

This paper focuses on point-to-point interface between the sentinel and the mobile phone user. Figure 2 shows the block diagram of MicroBaseJ by highlighting the interface between the sentinel and mobile phone. Data Request asks the data from sensors at a sample rate (for example every hour) and data is saved in the object and ready to send. Data Transfer sent the object of collected data to the user's mobile phone. The remote phone receives the notification from Sentinel as background in Data receive function and Data Storage function stores it into the Perst Lite database. The specified data will be look up from the database in terms of the user requirements and ready to present in Data Present function. Sample Rate Setting provides a menu to user to reset the sample rate and send to the Sentinel. The sentinel will store these data in Sample Rate Storage function. Same as Alarm setting and Storage.

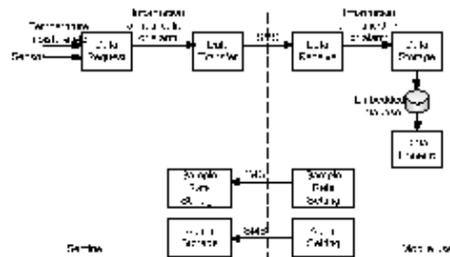


Figure 2. The Logical Diagram of MicroBaseJ

B. MicroBaseJ Architecture

The Cellular Sentinel is one of the pocket based informatics for asset management and remote control solution of preference. The Cellular Sentinel uses TINI DS80C400 and Wavecom FASTRACK as the hardware.

The TINI reference board is based on the DS80C400 processor [15]. The DS80C400 includes 1M RAM, 1M flash, 144-Pin SODIUM, and networking includes 1-Wire, CAN, serial, 10/100 Ethernet connections.

The TINI provides physical connectors to interface the TINI with other device such as Ethernet network, a serial device and a 1-Wire network. It provides a friendly interface for remote asset management and control systems. The developers can choose difference language, such as Java, C or even coding in 8051.

The Wavecom FASTRACK modem provides instant wireless capabilities. Housed in a rugged metallic casing, the

FASTRACK modem is built to withstand the toughest environments. Its open interfaces and AT commands can embed and run the application right on the WISMO platform and it is GSM/GPRS enabled.

The Sentinel is the product of Mobile Control Solutions (MCS) Ltd. N.Z. which uses this platform and others to collect information from electronic sensors and send to the mobile user.

MicroBaseJ intends to use Short Message System (SMS) as communication protocol to transmit collected data to user handset.

C. IDE and Software

Eclipse, an open source IDE (Interactive development environment) is chosen as the development environment. Its project is focused on creating an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software lifecycle, including enterprise development, embedded and device development, rich client platform, rich Internet applications, application frameworks, application lifecycle management and service oriented architecture. The Eclipse version of this project is Eclipse DSK (Development Software Kit) 3.2.2.

The developed language, Java Platform Micro Edition (J2ME), provides a flexible and robust environment for applications based on mobile and other embedded devices—mobile phones, PDAs (personal digital assistants), TV set-top boxes, and printers. J2ME contains elastic user interfaces, robust security, built-in network protocols, and support for networked and offline applications that can be downloaded dynamically. J2ME applications can be portable across many devices.

The Sun Java Wireless Toolkit (formerly known as J2ME WTK) is a set of tools for generating Java applications that run on devices obedient with the Java Technology for the Wireless Industry (JTWI, JSR 185) specification and the Mobile Service Architecture (MSA, JSR 248) specification. It includes built-in tools, utilities, and a device emulator. This project adopts WTK2.5. The Sun Java Wireless Toolkit 2.5 contains all advanced development features found in version 2.2, 2.3 Beta, 2.5 Beta 2, for example, MIDlet signing, certificate management, integrated over-the-air (OTA) emulation, push registry emulation, etc. It provides four simulators for user. The DefaultColorPhone is chosen as the simulator in this project

IV. DATABASE PERFORMANCE EVALUATION AND ANALYSIS

The key area of this project is the implementation of embedded database. Insert function, search function, delete function and modify function are the major features in database. The behaviour of search function, delete function and modify function are similar, performance could be measured through the search function, so this article focuses on the search function and insert function. The search

function includes sequence and random search. The following part compares Perst Lite and RMS. It is based on both simulation and real device. The comparison is based on the Java platform jre1.6.0_01 and Eclipse DSK 3.2.2.

A. Embedded Database Performance comparison based on simulation

The first comparison is based on Sun Java Wireless Toolkit 2.5 DefaultColorPhone. It compares Perst Lite to RMS based on Insert function, Sequence Search and Random Search. The executing time is the average for each record during the whole evaluation.

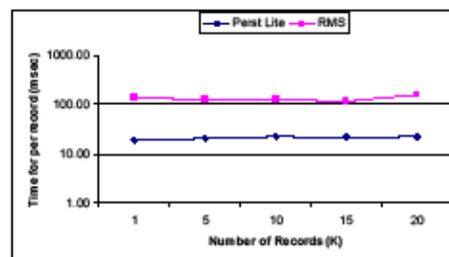


Figure 3. The Insert Function Performance Comparison

Within the Insert function, the code sets Perst Lite pool size as 64K, 100 records as one transaction. As can be seen, Figure 3 shows that the Perst Lite used around one tenth running time of RMS for Insert Function. The major reason is that Perst Lite used 100 records per transaction, this can really reduce the Input/Output (I/O) overhead. Another might be that the performance of Perst Lite is better than RMS.

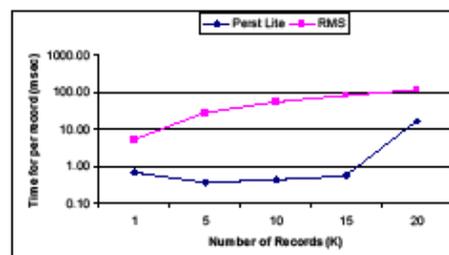


Figure 4. The Sequence Search Performance Comparison

Figure 4 compares Perst Lite and RMS in Sequence Search. RMS put all records into enumerator and search from the beginning to the end. Perst Lite set all records into iterator and go through them. The result shows that the performance of Perst Lite is better than the RMS. The searching time for RMS keep increase, Perst Lite keep stable

when record number less than 16000, after that the searching execution time rises significantly. The reason for this might be due to the search engine of Perst Lite.

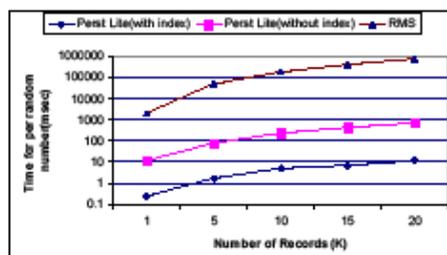


Figure 5. The Sequence Search Performance Comparison

Within Random search, there are 500 random numbers generated by program, RMS would retrieve all records by each random numbers. The run time is the average for each random number. Perst Lite did the same search with index and without index. Figure 5 shows that Perst Lite need less time to locate the random number than RMS. Furthermore, the search without index takes about 10 times more than that with index.

Table 2 Database Size Comparison

Record numbers (K)	Perst Lite (KB)	RMS (KB)
1	282	111
5	1225	556
10	2530	1112
15	4024	1677
20	5141	2241

Table 2 compares the database size of Perst Lite to RMS based on Sun Java Wireless Toolkit 2.5 DefaultColorPhone simulation. The record length is about 110 bytes. As can be seen, the data size of Perst Lite need twice of RMS. It means that Perst Lite needed more than 100 bytes in each record for the database overhead.

B. Embedded Database Performance comparison based on cell phone

The second part evaluation is based on the mobile phone, Nokia N73 with 42M memory and extra 1G miniSD card. The sized of processed data were limited to 11000 records. Figure 6 demonstrates the comparison of Insert Function between Perst Lite and RMS. It shows that both Perst Lite and RMS take more running time with the increase of record number. Perst Lite reflects better performance than RMS.

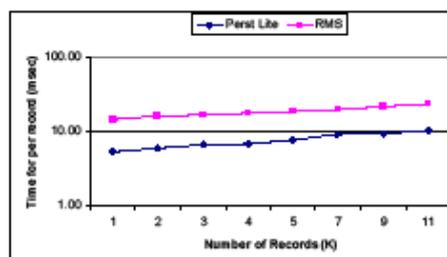


Figure 6. The Insert Function Performance Comparison

The Comparison of sequence search is illustrated in Figure 7. It seems that Perst Lite and RMS performance data are quite close. There is a significant difference with the result based on the simulation. The reason may be that the database was stored into the mobile phone memory rather than the disk (as is the case with the simulation). As such, it can reduce the I/O operation significantly.

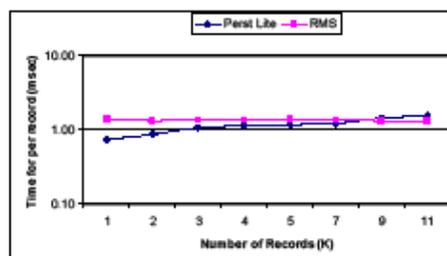


Figure 7. The Sequence Search Performance Comparison

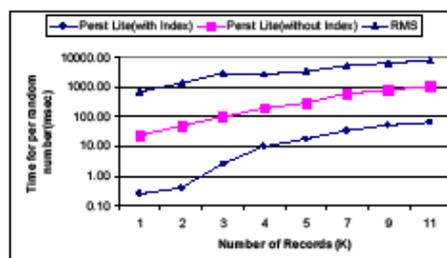


Figure 8. The Sequence Search Performance Comparison

Figure 8 illustrates the comparison of random search performance of RMS and Perst Lite with index and without index. The code generates 1000 random numbers and calculates the average timing for each random search. It shows that Perst Lite performance is better than RMS although all data stored in memory. The random search with

index needs less than one-tenth time of the search without index.

C. Performance Comparison Summary

As can be seen, through simulation, Perst Lite performance is much better than the RMS in insert function, sequence search and random search. The first explanation is that the Perst Lite has index which can locate records quickly. Secondly, the Perst Lite uses 64K as pool size. These can really reduce I/O operations then decrease the access time.

In contrast, on Nokia N73, the performance of RMS improves markedly, sequence search of Perst Lite are similar to RMS. Insert function of RMS is just more than twice slower than that of Perst Lite. The major reason might be that the database is stored in the phone memory rather than extended memory. There are no I/O operations when manipulating records. However, on Nokia N73, the performance difference on random search with and without index between Perst Lite and RMS is still significant. The reason is due to there is index in the Perst Lite.

Table 3. Perst Lite and RMS timing for per (based on 5000 records, msec)

Case		Perst Lite	RMS	
Simulation	Insert function	20.15	127.69	
	Sequence search	0.39	27.53	
	Random search	with index	2.2	44943.98
		without index	72.1	
Nokia N73	Insert function	7.49	18.28	
	Sequence search	1.17	1.4	
	Random search	with index	18.12	3305.15
		without index	286.34	

V. CONCLUSION

This paper reviews the mobile computing with embedded databases. The focus is on the Point-to-Point mobile embedded database. Four embedded databases were explored. These are Perst Lite, PointBase, db4o and RMS. Both the Perst Lite and RMS performance were evaluated. The evaluation covered evaluating the performance of the Insert function, Sequence Search and Random Search. Perst Lite reflects good performance and supersedes RMS in all of the above functions.

ACKNOWLEDGMENT

The authors would like to thank Hong Zhang and Murray McGovern from Mobile Control Solutions Ltd. N.Z. for

technical and project support.

REFERENCES

- [1] Mobile and World, "Subscriber statistics end Q1 2007," 2007.
- [2] S. Ravi, M. Chatbi, and H. Prasanna, "WAP AND SMS BASED EMERGING TECHNIQUES FOR REMOTE MONITORING AND CONTROL OF A PROCESS PLANT," in *ICSP'04. 2004 7th International Conference*, 2004, pp. 2672-2675.
- [3] K. Hung and Y. Zhang, "Implementation of a WAP-Based Telemedicine System for Patient Monitoring," *Information Technology in Biomedicine*, vol. IEEE Transactions on vol. 7, pp. 101-107, June 2003.
- [4] M. Nikolova, F. Meijs, and P. Voorwinden, "Remote Mobile Control of Home appliances," *Consumer Electronics*, vol. 49, pp. 123-127, Feb 2003.
- [5] S. Rahman and S. Bhalla, "Supporting Spatial Data Queries for Mobile Services," in *ACM International Conference on Web Intelligence*, 2005.
- [6] V. Lee, K. Lam, and W. Tsang, "Transaction Processing in Wireless Distributed Real-time Databases."
- [7] C. Yang and M. Kou, "Remote Monitoring and Control of PC Clusters Using Mobile Phones with J2ME," 2005.
- [8] J. Nichols and B. Myers, "Controlling Home and Office Appliances with Smart Phones," *PERVASIVE computing*, pp. 60-67, 2006.
- [9] F. Siegemund and C. Florkemeyer, "Interaction in Pervasive Computing Settings using Bluetooth-Enabled Active Tags and Passive RFID Technology together with Mobile Phones," in *The First IEEE International Conference on Pervasive Computing and Communications*, 2003.
- [10] G. Nikolakopoulos, M. Koudourakis, and A. Tzes, "An Integrated System based on WEB and/or WAP Framework for remote Monitoring and control of Industrial Processes," in *International Symposium on Virtual Environments, Human-Computer Interfaces, and Measurement Systems*, Lugano, Switzerland, 2003.
- [11] McObject, "McObject Releases Perst Lite, an Open Source Embedded Database For Intelligent Devices on Java's J2ME Platform," 2006.
- [12] DataMirror, "POINTBASE MICRO: The Ultra-Compact Database for Enterprise Mobility," 2004.
- [13] db4objects, "db4o Open Source Object Database."
- [14] McObject, "Perst Documentation," 2006.
- [15] Maxia, "Getting Started with TINI"

Appendix IV – The CD contents

There is a CD attached. It contains the thesis, the conference paper for ATNAC'07, and the code of the application.

A. The Code Introduction

The code includes the 7 classes, ReceiveSMSMsg, MainMenu, HorticultureData, pushProcess, MyLineCanvas, MyBarCanvas, and MyLineBarCanvas. These codes can be run on the

The ReceiveSMSMsg is the main class in the application. It offers an entry to the application for the mobile user. It implements major functions of the application, such as receiving the message from the Sentinel, storing the relevant information into the mobile database, and providing the interface to the MainMenu class.

The MainMenu provides an interface to the user to manipulate relevant operations. It allows the user to choose different functions, such as presenting the user with specific information on the screen, updating the alarm settings and sample rates, and sending the modified sample rate back to the Sentinel. It also provides an entry to one of the MyLineCanvas, MyBarCanvas, and MyLineBarCanvas.

The class pushProcess, deals with the message from the Sentinel. This object will be activated by the ReceiveSMSMsg object as a background process when the message arrives from the Sentinel. The HorticultureData is the class defined objects for the database. It has insert and read functions, and defines all attributes for the database, including time stamp and another ten items from the sensors. It is used for the ReceiveSMSMsg and the MainMenu classes.

The classes; MyLineCanvas, MyBarCanvas, and MyLineBarCanvas; are used for drawing different types of figures, such as the line chart, the bar chart, and their combination.

B. The Operating Environment

The application can be run on the computer with software such as JAVA JDK (Java Development Kit) (jdk-6u2-windows-i586-p), Eclipse (eclipse-sdk-3.2.2-win32), EclipseME (eclipseme.feature-1.7.8), J2ME Toolkit (sun_jave_wireless_toolkit-2_5_1), J2ME Polish (de.enough.mepose.ui_0.7.1), and Perst Lite (perst269).

This project adopts the Sun Java Wireless Toolkit, WTK2.5.1. The Eclipse version of this project is Eclipse DSK (Development Software Kit) 3.2.2.