DEVELOPING A PC CONTROLLER APPLICATION USING BLUETOOTH TECHNOLOGY

Malek Ali Ahmad Fehaid

UNIVERSITI UTARA MALAYSIA

2008

DEVELOPING A PC CONTROLLER APPLICATION USING BLUETOOTH TECHNOLOGY

A thesis submitted to the Graduate School in partial fulfillment of the requirements for the degree Master of Science (Information Technology)
Universiti Utara Malaysia

By

Malek Ali Ahmad Fehaid (800300)

KOLEJ SASTERA DAN SAINS
(College of Arts and Sciences)
Universiti Utara Malaysia

PERAKUAN KERJA KERTAS PROJEK
(Certificate of Project Paper)

Saya, yang bertandatangan, memperakukan bahawa
(I, the undersigned, certify that)

MALEK ALI AHMAD FEHAID
(800300)

calon untuk Ijazah
(candidate for the degree of) **MSc. (Information Technology)**

telah mengemukakan kertas projek yang bertajuk
(has presented his/her project paper of the following title)

DEVELOPING A PC CONTROLLER APPLICATION USING BLUETOOTH TECHNOLOGY

seperti yang tercatat di muka surat tajuk dan kulit kertas projek
(as it appears on the title page and front cover of project paper)

bahawa kertas projek tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan.
(that the project paper acceptable in form and content, and that a satisfactory knowledge of the field is covered by the project paper).

Nama Penyelia Utama
(Name of Main Supervisor **ASSOC. PROF. DR. WAN ROZAINI SHEIK OSMAN**

Tandatangan
(Signature) : Rozaini Tarikh (Date) : 11/11/08

ASSOC. PROF. DR. WAN ROZAINI SHEIK OSMAN
Director
ITU-UUM ASP COE
For Rural ICT Development
Information Technology Building
Universiti Utara Malaysia

Nama Penyelia Kedua
(Name of 2nd Supervisor): **MR. KHUZAIRI MOHD ZAINI**

Tandatangan
(Signature) : Tarikh (Date) : 11/11/2008.

GRADUATE SCHOOL
UNIVERSITI UTARA MALAYSIA

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a postgraduate degree from the Universiti Utara Malaysia, I agree that the University Library may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner in whole or in part, for scholarly purposes may be granted by my supervisor(s) or in their absence by the Dean of the Graduate School. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part should be addressed to:

Dean of Graduate School

Universiti Utara Malaysia

06010 UUM Sintok

Kedah Darul Aman.

ABSTRACT

Wireless communications have become more common and reliable to use in many applications, and is still developing. There are many types of wireless network connection, and one of these is Bluetooth technology. The Bluetooth technology supports many features that replace wires with low cost and high speed transmission. It is difficult to manage the networked PCs manually; the best solution for this problem is an easier way to manage a network remotely. *PC Controller* is a program to control the network by using messages through Bluetooth technology. This study concerns primarily about how to design and develop a *PC Controller* Application.

ABSTRAK

Komunikasi tanpa wayar semakin menjadi suatu kebiasaan dan boleh dipercayai untuk digunakan dalam banyak aplikasi, dan ianya masih dalam proses pembangunan. Terdapat banyak jenis sistem perhubungan tanpa wayar, dan salah satu daripadanya ialah teknologi "Bluetooth". Teknologi "Bluetooth" mampu untuk melaksanakan pelbagai perkara yang menggantikan wayar dengan kos yang rendah dan kelajuan pemindahan maklumat yang tinggi. Ianya adalah suatu yang sukar untuk menguruskan PC yang telah disistemkan secara manual; penyelesaian yang terbaik untuk masalah ini adalah suatu cara yang lebih mudah untuk menguruskan suatu sistem adalah secara kawalan jarak jauh. Pengawal PC merupakan suatu program yang mengawal sistem dengan menggunakan maklumat melalui teknologi "Bluetooth". Kajian ini mengambil berat terutamanya tentang bagaimana untuk merangka dan membina sebuah pengaplikasian kawalan PC.

ACKNOWLEDGEMENT

Firstly I would like to state my sincere appreciation to my supervisors Assoc. Prof. Dr. Wan Rozaini Bt Sheik Osman, and Mr. Khuzairi Mohd Zaini for their patient guidance, encouragement, understanding, and excellent advice throughout this study.

I am also thankful to all my colleagues and friends at UUM, especially from the faculty of information technology for their help and support, with whom I shared pleasant times.

I would like to dedicate this work to my father and to the memory of my mother

Finally, I am deeply and forever indebted to the people in my life who touched my heart and gave me strength to move forward to something better. The people who inspire me to breathe, who encourage me to understand who I am, and who believe in me when no one else did. I am deeply devoted to my adoring fiancé, for the patience and confidence she had in me, throughout the times I was away from her.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF CHARTS

CHAPTER 1

INTRODUCTION

1.1 Introduction

PC controller is a program to control the computer by using messages through Bluetooth technology. In *PC Controller*, there is a need to support the system with Bluetooth (embedded or USB dongle), to control the computer by sending files with certain names to the computer to take the desired action. The process starts when a user sends a message from his/her mobile phone to a target PC. After the message reaches at the destination it will trigger that specific file, while the *PC Controller* checks the arriving message according to their name and from there action will be taken.

The most common type of network connection is the wireless connection. However, Bluetooth technology supports many features that can replace the traditional wired network connections with a low cost, high speed transmission, low power, secure and robust standard for short range connectivity. The technology has been designed for ease of use, simultaneous voice and data and multi-point communications. It supports a range of 10 m, which can be increased up to 100 m with the use of an amplifier (Rumiana. et al., 2005).

Bluetooth technology exists in many devices, such as telephones, printers, modems and headsets. Bluetooth technology is commonly used to transfer sound data for example Bluetooth mobile headset and it used to transfer the data hand-held computers (Dursch. et all. 2003). The Figure 1.1 shows some devices used Bluetooth technology.

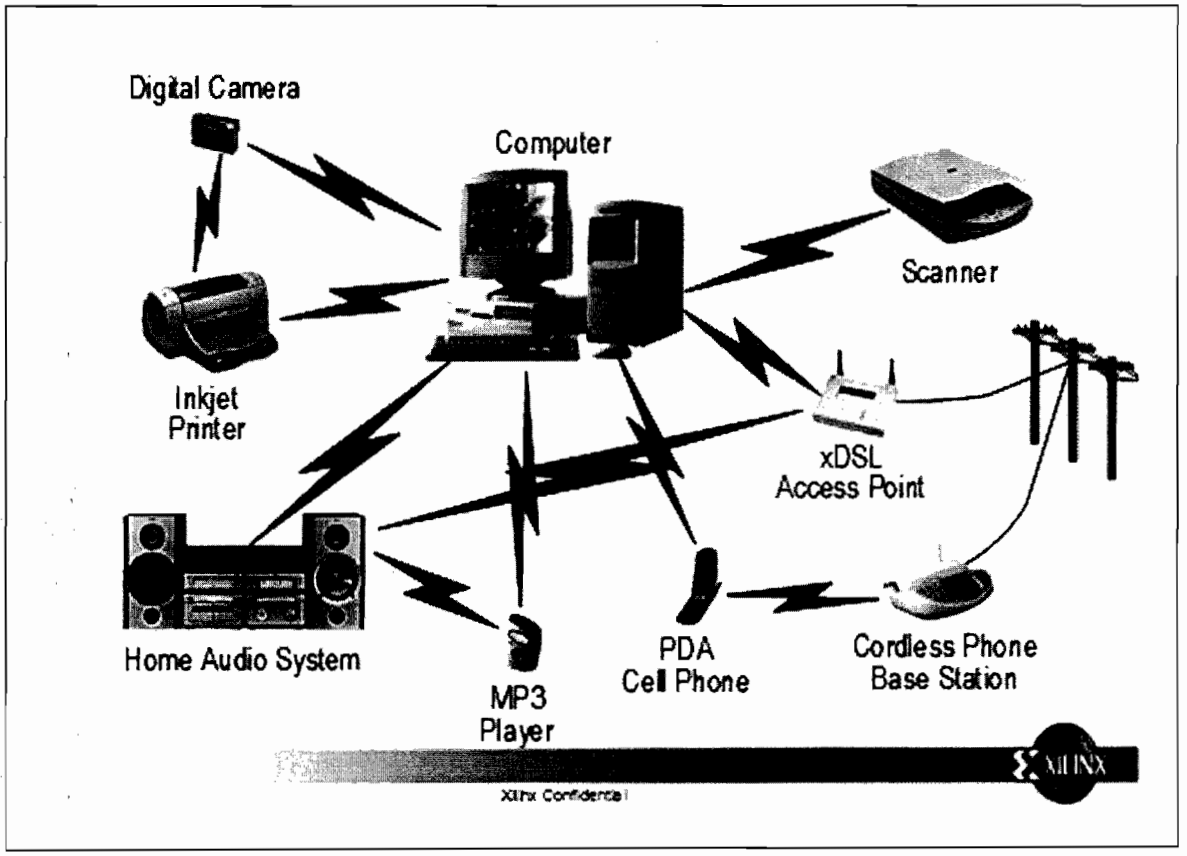

Figure 1.1: Using Bluetooth (Sand, 1999)

This study is proposed for the design and implement of a *PC Controller* application that aims to execute computer command remotely from a distant or another place over networked PCs.

1.2 Problem Statement

Suppose students leave the computers without shutting it down. Then the lab assistant must manually shutdown all the PC in the lab, if that was not done then there will be a waste of electricity and computer usage. In addition if the lab assistant wants to check the network connection continuously. Another scenario that may happen is someone may forget to logoff the PC, when he/she leaves his/her device unprotected and unsecured. It is inconvenient to go back to office to logoff the PC.

These kinds of problems should solve effectively. It is inconvenient to shutdown check the network connection, and logoff all the networked PCs manually this is not the optimal solution. It is time consuming, difficult and it seems that it is much more complicated to control multiple devices.

1.3 Research Questions:

Some of the questions that must be answered through the research are:

1. What is the current method for managing the labs and the networks of the departments?
2. What are the requirements of PC controller Application?
3. How to control more than one PC by using Bluetooth technology?

1.4 Objectives

The main objective of this study is to design and develop a *PC controller* by using Bluetooth technology that executes some of the computer commands from a distant or another place.

The sub objectives are:

1. To identify the requirements of the proposed system.
2. To design the PC application
3. To develop the *PC controller* system

1.5 Scope

The scope of this study is to design and develop a *PC controller* application that enables the lab assistants to control one or multi devices such as one or more PCs at the same time through using Bluetooth technology in *UUM* labs environment.

This includes:

1. Shutdown all the networked PC.
2. Check the network connection.
3. Logoff.
4. Send text messages and open it.

1.6 Significance

This study aims to solve all the problems mentioned before by designing and developing of a *PC controller*. The major benefits that can be seen through operating this system would include institutional security; the departments in *UUM* can get benefits by using this system to gain control over a network of various PCs in such cases like when some employees may forget a password of their PCs. Also lab assistant can send Attention to the student inside the lab to inform them to leave the lab, so by using this system the network can be controlled easily by sending one order at the same time to the whole network. As for lecturers they might be able to send instructions such as question to all the students in the lab.

1.7 Report Structure

Chapter two presents a review on the literature of controlling the networks and previous technologies works, from another side this study presented the wireless communication and focus on the Bluetooth technology and his details.

Chapter three research methodologies implemented in order to achieve the objectives that are used in the four or five chapters are focused on.

Chapter four discusses the *PC Controller* design issues and implementation using the methodology, and highlight on the evaluation techniques and the testing performed and its results.

Chapter Five describe the research Conclusions and Future Work.

1.8 Summary

Controlling one computer or more by using Bluetooth technology provides a solution to manage a network without wires, with low cost and high speed manner. Nowadays there are many of devices which it is used to do the same things but the problem it is so expensive, this project aim to solve this problem through using mobiles Bluetooth. The deferent between use mobile Bluetooth and previous device is using the messages through the Bluetooth, because now days every person has mobile at least, so it used by all. *PC Controller* is simple, but it has powerful usage, it will be useful in academic field, for example: to deal with beginners students, and later can de developed to be more flexible and reliable.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The solution of this study is seen as a simple application that can provide a command that will execute many tasks to the network with the help of using Bluetooth technology. This study presented a review on some of application used to control the network to get some information to make a solution and understand how the previous technologies works, from another side this study presented the wireless communication and focus on the Bluetooth technology and his details.

2.2 Implementation and Design of PC Control System Using Mobile-Based Database

This system implemented to control the computer system connecting to PC with mobile devices. This system is designed by mobile system based and database, and mobile remote control system for managing PC via the mobile connection that applies IP address and database. PC Control System implemented to manage the multimedia (music, image and movie files) and Shutdown actions through the connection between mobile device and PC via wireless network, and web service for wire network (Lee and Bang, 2006).

2.2.1 System Overview

Mobile device or web request the access authorization to PC, and by middle database server allow the user of the mobile or the web to connect to the PC. This system uses specific access method it is IP address of PC. There is a main procedures to use this system; at the first checks the login information from the user, if this information right the user can receive access authorization to client PC else the user cannot access to client PC. Figure 2.1 shows the system overview (Lee and Bang, 2006).

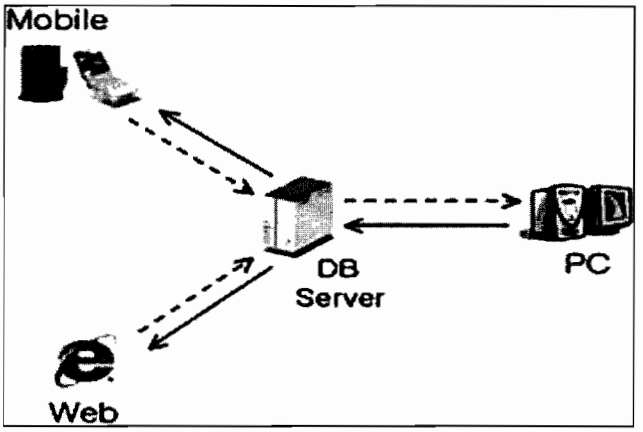

Figure 2.1: PC Control System Overview (Lee and Bang, 2006).

The processing of the control system will done by many steps; at the first the control system will manage users and file, then the database system will check the login information through users Table and activates program when mobile device or web request login. Figure 2.2 shows the user and file process (Lee and Bang, 2006).

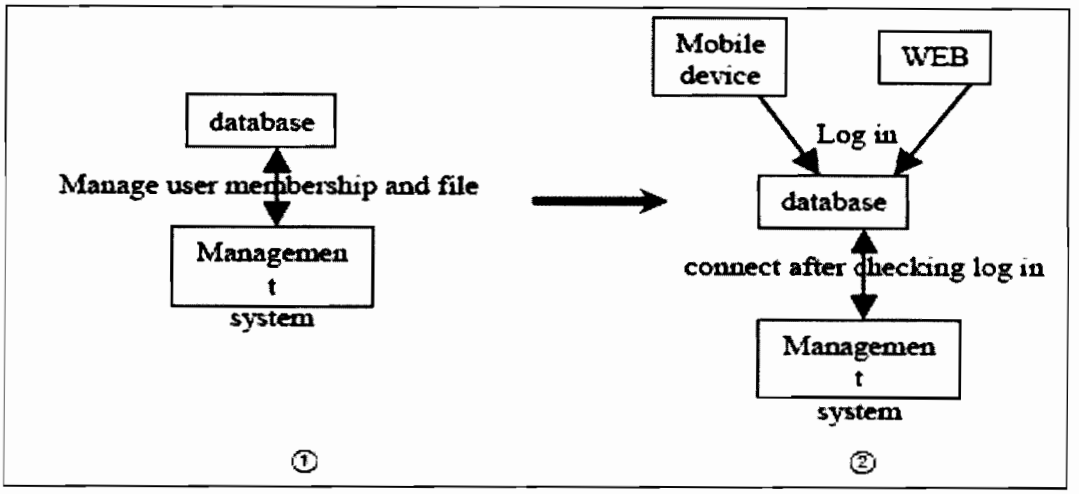

Figure 2.2: Processing of User and File (Lee and Bang, 2006).

After the processing, program is activated and mobile device and web program can request program running. When requesting program, user can manage private program. Database receives the requested program list and run a program, as it shown in the program operating in Figure 2.3 then the user and the file management must activate the system program. Then there is a process to register a user and login after starting a system (Lee and Bang, 2006).

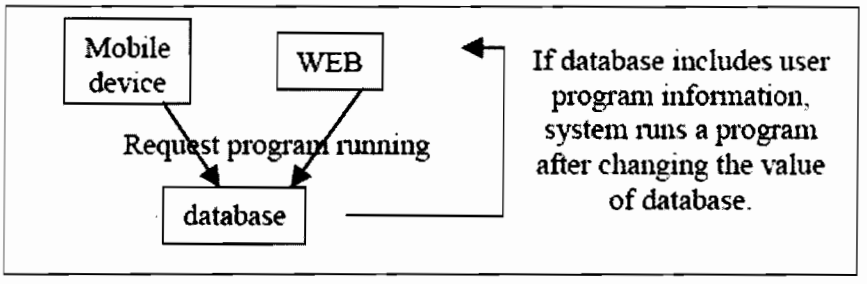

Figure 2.3: Program operating (Lee and Bang, 2006).

2.2.2 The Management System

The control system determines management system to RemoteClient and checks database by login information. Next, the control system runs the program conformed to the data of database or exits system (Lee and Bang, 2006).

2.2.3 WEB

WEB program connected to database. The login page in this program it is first page to checks ID and password at login, and generates a session for log in step on web. Then logout page removes a session. The user will input ID and password in the login page, and this information is passed to Remote Control page then this page shows the list of program and system managed by a user. User selects the menu of start and download, and sees the progressing of transmission (Lee and Bang, 2006).

2.2.4 Mobile

The mobile web page checks login page, ID and password, and the next page indicates the program and computer information to manage, and transmits the value. The next page starts a program with the value from the second page. This page is for system off (Lee and Bang, 2006).

2.2.5 Database

Table s in the system database include user information and start program does program information and there is a Table to support system off and can exit the system with ID and the value of column as system down. Database can manage private program (Lee and Bang, 2006).

2.3 PC Remote Shutdown Pro 5.0

PC Remote Shutdown Pro 5.0 is a Shutdown Software, and it can instantly or automatically power up, shutdown and controls the networked PCs from a central PC running *the PC Remote Shutdown Program Server* software. With one button press, by using this program can power up, shutdown, reboot, log out, lock up, run the screen saver or turn off the monitors of those PCs in the network that are running the *PC Remote Shutdown Program Client* software (Softsea, 2006).

PC Remote Shutdown Client is designed to work well on multi-users workstation. It can be controlled by the server program even when there is no logon or when the client PC is locked up. This software designed for many environment and scenarios such as remote shutdown and control of PCs in a PC network gaming shop or internet cafe or cyber cafe (Softsea, 2006).

2.4 Personal Area Networking (PAN)

Looking back the literature in this area there is one of the last usage models and it is the *Personal Area Networking* (PAN), focuses on the ad-hoc formation and breakdown of personal networks, see Figure 2.4. PAN can be both wired and wireless. For wireless PAN an example of it can be PICONET, which is PAN using Bluetooth connection (Rabaey, et al., 2000).

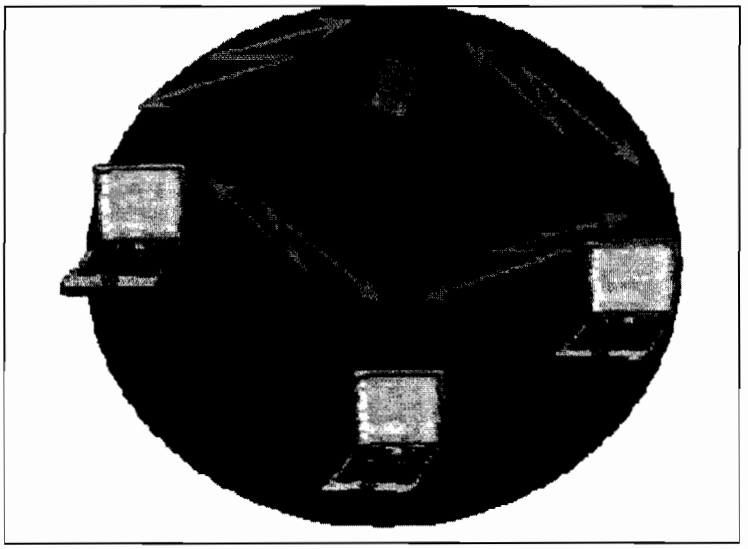

Figure 2.4: Personal Area Networking. (James, 2000)

Imagine meeting someone in an airport and quickly and securely exchanging documents by establishing a private PICONET. Think about people getting important information and data easily, quickly and efficiently. It is expected that in the future, Bluetooth kiosks could provide access to electronic media that could be quickly downloaded for later access on the mobile device (James, 2000).

2.5 Wireless Communication

Wireless communications is one of the most active areas of technology development today, and it is a fast growing part of the communications, with the possibility to provide high-speed, high-quality information exchange between portable devices located anywhere in the world (Samppa,1999). Potential applications enabled by this technology include multimedia internet-enabled cell phones, smart homes and appliances, automated highway systems, video teleconferencing and distance learning, and autonomous sensor networks, to name just a few. However, supporting these applications using wireless techniques poses a significant technical challenge (InformIT, 2003).

2.5.1 Current Usage

Wireless is a way of communication and it uses low-powered radio waves to send and receive data between devices and it removes the cables or cords. Common uses include the wireless networking of computers and cellular mobile phones. These electromagnetic waves travel through the air at almost the speed of light. Frequency is measured in Hertz (cycles per second). Radio frequencies are measured in kilohertz (KHz or thousands of cycles per second), megahertz (MHz or millions of cycles per second) and gigahertz (GHz or billions of cycles per second). In general, signals with longer wavelengths travel a greater distance and more easily penetrate through and around objects than signals with shorter

wavelengths. High-powered transmission sources usually have need of government licenses to conduct on a particular wavelength. Low-powered conduction sources, for example those used in Wi-Fi networking to send and receive data, are repeatedly unfettered (Verint, 2007). Figure 2.5 show some of devices used wireless network.

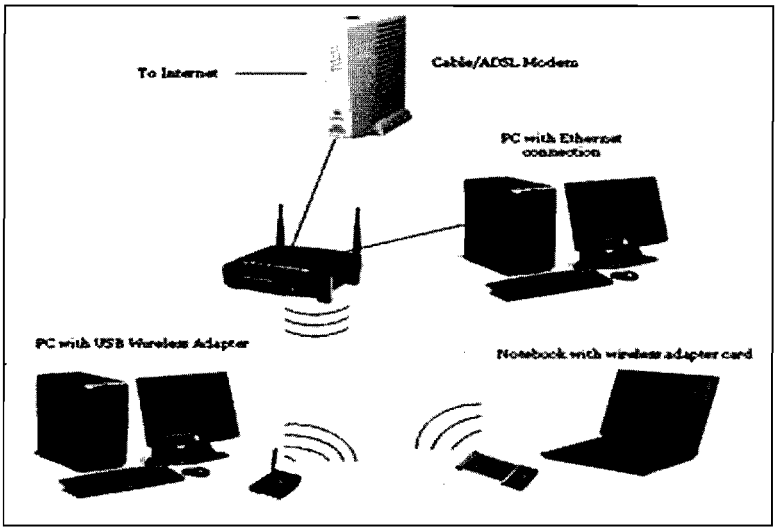

Figure 2.5: Wireless Network (Magnuslator, 2007)

2.5.2 Wireless Standards

The following is some of the wireless standards and their details:

2.5.2.1 IEEE 802.11 (Wi-Fi)

It is not a set of wireless LAN standards developed by working group 11 of IEEE 802. The term is also used specifically for the original version; to shun the perplexity that is sometimes called "802.11legacy" (Point, et al, 2005).

The 802.11 family at this time includes three split protocols that focus on encoding (a, b, g); security was originally integrated, but is now part of other family standards (e.g., 802.11i). Other standards in the family (c-f, h-j, n) are service enhancement and extensions, or corrections to previous specifications. 802.11b was the first widely accepted wireless networking standard, followed, paradoxically, by 802.11a and 802.11g (Zimmermann, 2004).

2.5.2.2 IrDA (Infrared Data Association)

IrDA is a wireless technology, which determines physical specifications and communications protocol standards within a short range exchange of data on the infrared light, for uses such as personal area networks (PANS) (John, et al, 2007).

Infrared wireless used for short, medium range communications and control some systems are working in line-of-sight mode, which means that have to be visually unhindered straight line through between the transmitter and receiver (John, et al, 2007).

In contrast the radiofrequency (RF) wireless communication, IR wireless can not pass through walls. Thus, infrared connection or control is generally not possible to different rooms in the house or between different

houses in the neighborhood (unless they have facing windows). This might seem like weakness, but IR wireless more private than RF wireless. Some IR wireless schemes offer the level of security comparable to that of hard-wired system. It is very difficult, for example, eavesdrop on well-engineered line-of-sight, infrared laser communications (John, et al, 2007).

2.5.2.3 DECT

DECT (Digital Enhanced Cordless Telecommunications) is a standard for digital handled phones, usually used for domestic or companies purposes. DECT be able to use for send and receive data for wireless. It is a cellular system. A main differentiation between the cellular systems is the cell radius; the radius in DECT cells from 25 to 100 meters (Silva, 2006).

2.5.2.4 DSRC

DSRC (Dedicated Short Range Communications) is a short to middle range communications service, it is intend to be a complement to cellular communications by providing very high data transfer rates in circumstances where minimizing latency in the communication link and isolating relatively small communication zones are important (Whyte, 2005).

2.1 Comparison of Wireless Technologies

There are many differences between Bluetooth technologies such as the connection type, range, data rate and the range, Table 2.1 shows a comparison between the some of wireless technologies.

Table 2.1: Comparison of Wireless Technologies

Feature / Function	Infrared	802.11 Wireless LANs	HomeRF	Bluetooth™
Connection Type	Infrared, narrow beam	Spread spectrum	Spread spectrum	Spread spectrum
Spectrum	Optical 850 nm	RF 2.4 GHz	RF 2.4 GHz	RF 2.4 GHz
Transmission Power	100mW	100mW	100mW	1mW
Data Rate	16Mbps	1Mbps, 2Mbps	1Mbps, 2Mbps	1Mbps
Range	1 meter	100 meters	Typical home	10 meters
Supported Devices	2		127	8
Voice Channels	1	VOIP	6	3
Addressing	32 bit physical ID	48 bit MAC	48 bit MAC	48 bit MAC

(Trap17, 2006)

2.2 Bluetooth Technology

2.2.1 Bluetooth Networks

Bluetooth allowed devices as of ad-hoc networks, when they come into contact with each other. These networks built through using scatternet and *PICONET* structure algorithms (Law. et al, 2001).

From 1 to 7 devices can become a piconet. One device is designated to be master, by each device generating random numbers determining whether the device will perform to be master and request slaves or suppose to be a slave and scan for the master. Because only 7 nodes are allowed to participate in a piconet, scatternets are formed by linking many piconets together via shared slave nodes. The number of piconets to which a device belongs is termed its degree. A master node in a piconet may only have a degree of one, meaning a master node may not be shared between two piconets. The joint slaves are occasion completed between the piconets to which it belongs and data transfers between the piconets have to be transfer using the joint slave (Law, et al, 2001).

The time required to transfer data to be passed through joint slaves depended on the slave shared switching between piconets. Joint slaves need to

have timed come together points with piconet masters in arrange to exchange data (Misic, et al, 2004).

These ad-hoc properties of piconets and scatternets allow the data to be exchanged between a lots of Bluetooth devices at the same time as those devices are moving between piconets within a scatternet, well enabling the devices to physically move around although data are currently exchanged between them (Dideles, 2004).

Figure 2.6 shows *Ad Hoc Networks* such as Bluetooth network, IEEE 802.11 network and mobile phone network.

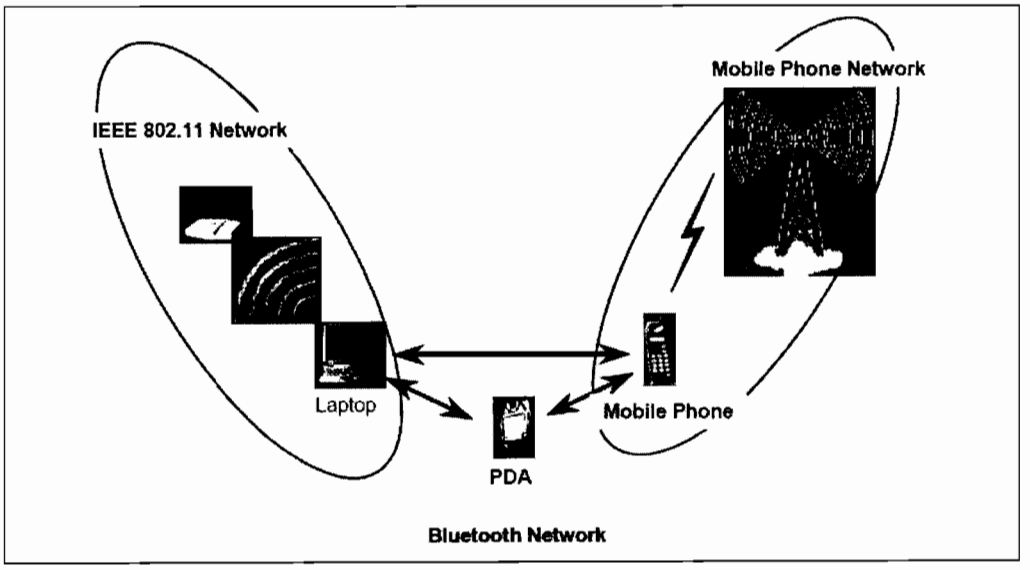

Figure 2.6: Ad Hoc Network (Karygiannis & Owens, 2002)

2.2.2 What is Bluetooth?

Bluetooth wireless technology is a short-range communications technology intended to replace the cables connecting portable and/ore fixed devices while maintaining high levels of security. The key features of Bluetooth technology are robustness, low power, and low cost. The Bluetooth specification defines a uniform structure for a wide range of devices to connect and communicate with each other (Rumiana. et al. 2005).

Bluetooth is a standard for short range, low power, and low cost wireless communication that uses radio technology. Over 2100 companies around the world already support Bluetooth technology. The wireless personal area network (WPAN) technology, based on the Bluetooth specification, is now an IEEE standard under the denomination of 802.15 WPANs. This work presents an overview about the Bluetooth communication (IEEE, 2005).

2.2.3 Bluetooth Architecture

The Bluetooth technology is divided into two specifications: the core and the profile specifications. The core specification discusses how the technology works, while the profile specification focuses on how to build interoperating devices using the core technologies (James, 2000).

Bluetooth designed and optimized for use on many devices such as mobile. PDA's, printers, desktops, keyboards, joysticks and basically any device provides a short range Bluetooth radios operating in the free 2.4GHz Industrial-Scientific-Medical (ISM) band integrated into them (single chip). It uses Frequency Hop (FH) spread spectrum, which divides the frequency band into a number of hop channels. Bluetooth radios use tiny radio-frequency transmitters, no larger than 1.0 by 0.5 inches that can run off a watch battery for months. Power considerations are always important for battery-powered mobile devices, and Bluetooth's low power modes meet those requirements with less than 0.1 W active power. And since Bluetooth is designed for both computing and communications applications, it is designed to support high quality simultaneous voice and data, with robust data transfer rates of up to 721 Kbps, it can use for both synchronous and asynchronous services and simple mixing of TCP/IP for the networking issues (Dasgupta, 2000). Figure 2.7 shows the Bluetooth Architecture.

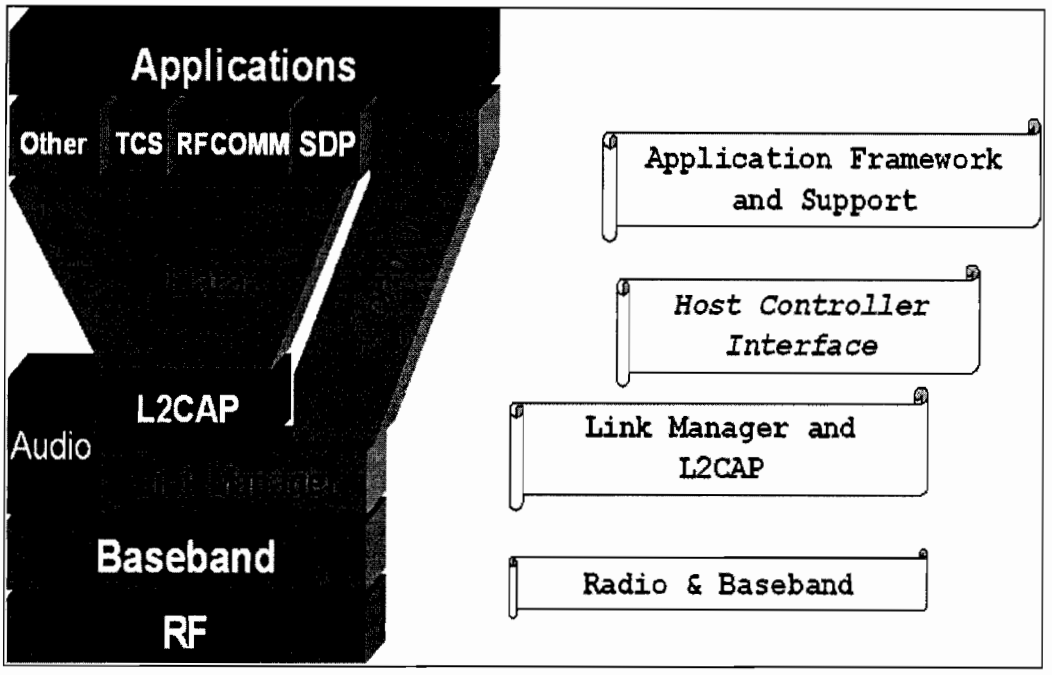

Figure 2.7: Bluetooth Architecture (Rathi, 2000)

2.2.4 Bluetooth Protocol

Bluetooth identify a protocol stack, the protocol stack allows hierarchical ad-hoc networks in the form of piconets, in which Bluetooth devices form themselves into point-to multipoint picocells of seven slaves under the control of one master. Multiple piconets in overlapping coverage areas form "scatternets". Although Bluetooth has been standardized for quite some time, Bluetooth devices are still not widely available. The Bluetooth devices which are currently available are only point-to point or point-to-multipoint devices. True ad-hoc multipoint Bluetooth devices are still yet to be commercialized (Song, 2001).

Protocol stack contains a radio layer which is it at the bottom of the Bluetooth protocol stack drown, which is it shown below, which forms the physical connection interface. The baseband and Link Manager Protocol (LMP) that reside over it are basically meant to establish and control links between Bluetooth devices. These three bottom layers are typically implemented in hardware. The Host Controller Interface (HCI) layer is required to interface the Bluetooth hardware to the upper protocol, which is Logical Link Control and Adaptation Protocol (L2CAP). The host controller is required only when the L2CAP resides in software in the host. If the L2CAP is also on the Bluetooth module, this layer may not be required as then the L2CAP can directly communicate with the LMP and baseband. Figure 2.8 shows the Bluetooth protocol stack (Sahbudin, et al, 2005).

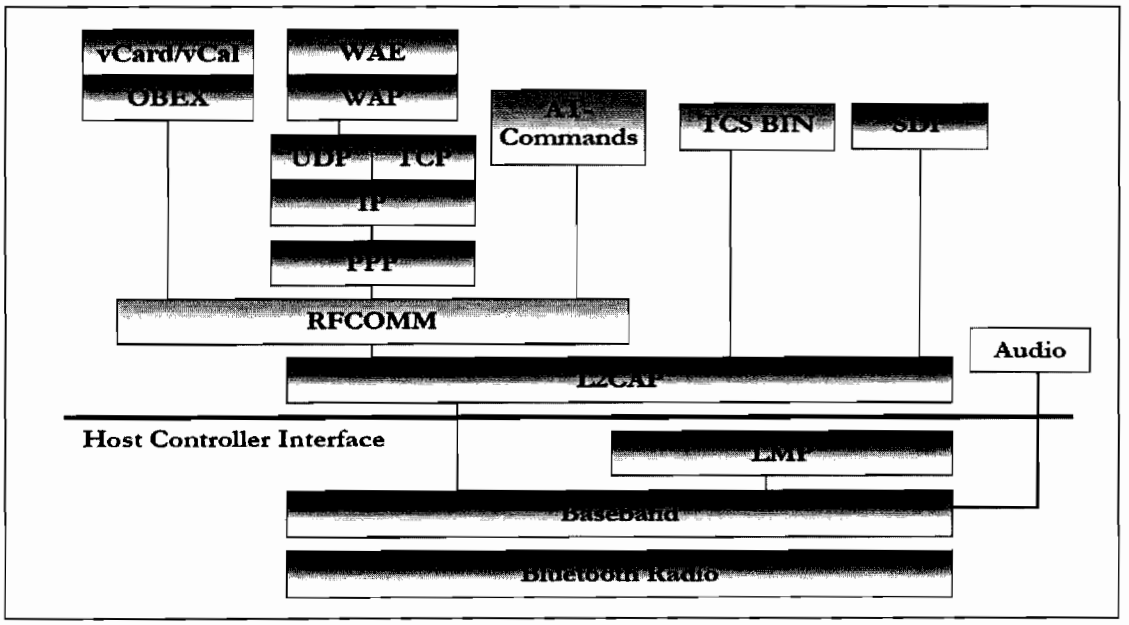

Figure 2.8: Bluetooth Protocol Stack (Jaquier & Drapel, 2005)

Bluetooth has many packets and each packet has a format, this format is a static, it is starts with a 72-bit access code. This is followed by a 54-bit header containing error correction, retransmission and control information. Finally the packet contains a payload of 0 to 2745 bits. Three methods, Forward Error Correction (FEC), Automatic Repeated Request (ARQ) and Cyclic Redundancy Checks (CRC), these used for check the errors and made a correction through Bluetooth (Nokia, 2003).

Link Manager Protocol (LPM) is a Bluetooth stack layer and this layer accountable for creation the links between Bluetooth devices and deals with master/slave switching, low power modes, clock offsets and packet size negotiation. This layer, although not critical to this project, also handles the exchange of authentication and encryption information. The Logical Link Control and Adaptation Protocol (L2CAP) allow complex of the protocols above it by segmenting and reassembling packets (Asahina, 2000).

The Service Discovery Protocol (SDP) is a Bluetooth stack layer and this layer allows Bluetooth devices to present and realize services. SDP passes bitmasks, representing advertised services, to all backbone nodes. This allows other devices to discover the type and location of a service on a Bluetooth network quickly (Nordbotten et al, 2004).

2.2.5 How Bluetooth Works

Bluetooth device work with other Bluetooth device, at the first the Bluetooth resaves the information that is carried by a wire and send this information at a special frequency to another Bluetooth device. Both the sending and receiving devices have what is called a Bluetooth receiver chip, which converts data into a wireless transmission and then reverse to normal again, conditional on if it is sending or receiving data (Iogear, 2000). Bluetooth can be used to connect almost any device to another device. Bluetooth can be used to form ad hoc networks of several (up to eight) devices, called Pico nets (Vainio, 2000).

When Bluetooth devices first connect, there is a piconet master that initiates the connection, and the others are slave devices. One piconet can have a maximum of seven active slave devices and one master device. All communication within a Pico net goes through the Pico net master. Two or more piconets together form a scatter net, which can be used to eliminate Bluetooth range restrictions (Haataja & Keijo, 2006). It is not possible to be a master of two different Pico nets because a Pico net is a group of devices all synchronized on a hopping sequence set by the master. For that reason, any devices that share a master must be on the same Pico net. Scatter net environment requires that different Pico nets must have a common device (so-called scatternet member) to relay data between the Pico nets (Haataja & Keijo, 2006).

Bluetooth wireless technology enabled devices operate in the unrestricted 2.4GHz, and the industrial, science and medical band ranges between 2.40GHz and 2.483GHz (Siegemund & Rohs, 2003).

Bluetooth wireless technology devices at random get between frequencies up to 1600 times per second much faster than other types of devices that use the industrial, science, and medical band. This means that if another device, such as a 2.4 GHz cordless phone interferes with a Bluetooth wireless technology network at a particular frequency, the interference only lasts for about 1/1600 of a second until the Bluetooth wireless technology devices hop to another frequency. This gives Bluetooth wireless technology networks a high immunity to interference from other 2.4-GHz devices. (HP, 2004).

There are three classes of Bluetooth wireless technology radio devices, each with a different maximum range: *Class 1* (100 meters); *Class 2* (50 meters); and *Class 3* (10 meters). HP notebooks feature *Class 3* BWT radios and HP printers feature *Class 1* radio (HP, 2004).

Table 2.2: Classes of Bluetooth wireless technology radio devices

Class	Range	Power consumption	Applications
1	Up to 330 ft (100m)	Greater	Typically used by devices that require extended range, such as Bluetooth to USB Adapters (IOGEAR GBU311) and more.
2	Up to 66 ft (20m)	Lesser	Typically used by devices which do not require great range and should conserve notebook battery power, such as Bluetooth MiniMice (IOGEAR GME225B), Bluetooth GPS (IOGEAR GBGPS201), Printer Adapters (IOGEAR GBP201), and more.
3	Up to 33ft (10m)	Least	Used by devices require very short range, such as cell phones, PDAs

(Iogear, 2000)

2.2.6 Bluetooth Technology Specifications

Bluetooth technology operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4 to 2.485 GHz, using a spread spectrum, frequency hopping, full-duplex signal at a nominal rate of 1600 hops/sec. The 2.4 GHz ISM band is available and unlicensed in most countries (Bluetooth, 2006).

Bluetooth devices within a 10 to 100 meters (or 30 to 300 feet) range can share data with a throughput of 1 Mbps for Version 1.2 and up to 3 Mbps for Version 2.0 + Enhanced Data Rate (EDR). Data is transmitted between

Bluetooth devices in packets across the physical channel that is subdivided into time units known as slots. As described in an article of JDJ, the world's leading java resource, the radio layer is the physical wireless connection. To avoid interference with other devices that communicate in the ISM band, the modulation is based on fast frequency hopping. Bluetooth divides the 2.4 GHz frequency band into 79 channels, 1 MHz apart (from 2.402 to 2.480 GHz), and uses this spread spectrum to hop from one channel to another, up to 1,600 times per second (Mikhalenko, 2006).

2.2.7 Bluetooth Security

The security in wireless technology is a main matter today. Wireless networks are essentially insecure. Any person who can cut off a radio transmission is able to take the data, get access or perform an ever-widening of evil deeds to a wireless network, and worse, to the users on that network. The security has become a priority for the industry and, in June 2004, the IEEE announced the adoption of the 802.11i security standard, which is aimed at vastly improving the overall security and privacy of wireless network data and SSIDs and deny access to illegal intruder (Price, 2007).

In generally speaking security issues in wireless communication has been the subject of great debate that. From technical perspective it can be viewed that wired networks are more secure than wireless. The wireless technology is

relatively new and there is a lot come from the road ahead. Therefore, Bluetooth is included in this security threat. The security concerns should concentrate on the vulnerabilities of the devices allowed participating in the Bluetooth networks. Specifically it is necessary to address security concerns for confidentiality, data integrity, and network availability. Anyway, it's possible to guarantee the security to the system.

Security plays an important role in the creation of Bluetooth. Bluetooth SIG has put much effort to make Bluetooth a secure technology and has security experts who give critical security information. In general, Bluetooth security is divided into three modes: One of them not secure, and service level enforced security, the last one is link level enforced security (Rodrin, 2006).

In non secure, a Bluetooth device does not initiate any security actions. In service level enforced security mode, two Bluetooth devices can establish a non secure Asynchronous Connection-Less (ACL) link (Hattar, 2006). Security procedures, namely authentication, authorization and optional encryption, are initiated when a L2CAP (Logical Link Control and Adaptation Protocol) Connection-Oriented or Connection-Less channel request is made (Rhodes, 2007).

2.2.8 Bluetooth Exchange Folder

The Bluetooth directory to another Bluetooth device has access. Devices that can access to a Bluetooth exchange folder, also devices can access to all subfolders in that folder and all files in these folders. By default, the Bluetooth exchange folder is placed in my documents. If you have the location of the Bluetooth exchange folder, not place it in the root directory or a system; moving the Bluetooth exchange folder to the root directory or folder system enables remote devices to damage the operating system on your computer. The Bluetooth exchange folder is shared by the post PIM transfer and file transfer services. The location of the folder can be specified by the properties page of one of these services. If the user has the location of the Bluetooth exchange folder on any of these services, the path is automatically updated to another department. The Bluetooth imaging service also uses this folder to store pictures taken (Broadcom, 2006).

2.2.9 Bluetooth Technology Strengths

The Strengths of Bluetooth technology summarize by a lot of strength characteristics that are missing in other technologies (Burns, 2006).

- Inexpensive
- Superior range compared with IR
- Designed for ad hoc network

- Both voice and data channels
- Well defined specification
- Large number of implementing companies. (Kansal, 2002)

2.2.10 Bluetooth Technology Weaknesses:

There are some of weaknesses in the Bluetooth technology.

- Slow communication rate (1 Mbps)
- Short networking range
- Limited number of network connections (Jakobsson and wetzel, 2000)

2.3 Technology Acceptance Model (TAM)

Technology acceptance model (TAM) expected wide support through validations, applications, and replications. TAM is the most powerful research model in the studies of the determinants of information technologies (IT) acceptance (Chau, 1996).

Users to accept and using the technology, the model TAM is the most cited. According to TAM, apparent usefulness and perceived ease of use are the primary inspiring factors for the adoption and use of new technologies. One of the apparent usefulness are using technology will bring better results are considered to have a

degree. Useful capacity is used being used advantageously. However, perceived ease of use a specific need to use the system, the degree of effort is about the recognition. In this case, the great difficulty and the freedom from work is the concept of ease. Perceived usefulness in TAM model at first refer to task related efficiency, performance, and effectiveness (Davis, 1989).

Figure 2.9 shows the TAM model, a key purpose of the TAM to give a basis for tracing the contact of external issues on internal values, attitudes and intentions (Davis. et al. 1989).

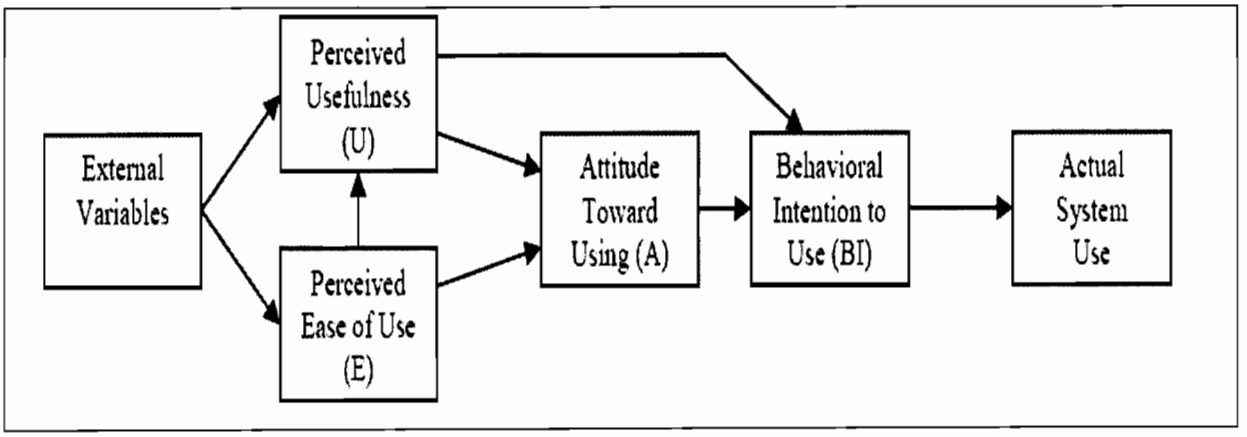

Figure 2.9: Technology Acceptance Model (Davis, et al, 1989).

2.4 Summary

. This chapter has discussed the some of application used to control the network and the wireless communication and focus on the Bluetooth technology. The next chapter will talk about the methodology of the research.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

Research methodology for the search is an organized way of nature and tries to solve a problem of research, not only a simple collection of the method (Kothari, 1985).

The main focused in this research is the methods, techniques and methods used by researchers at the time of completion for research. The research methodology in this study is a good method, a good selection, described and accepted in many researchers in the field of information, system design research (Vaishnavi & Lemoine, 2006).

This research was done in several steps. The Figure 3.1 shows the main stages in the research, design methodology.

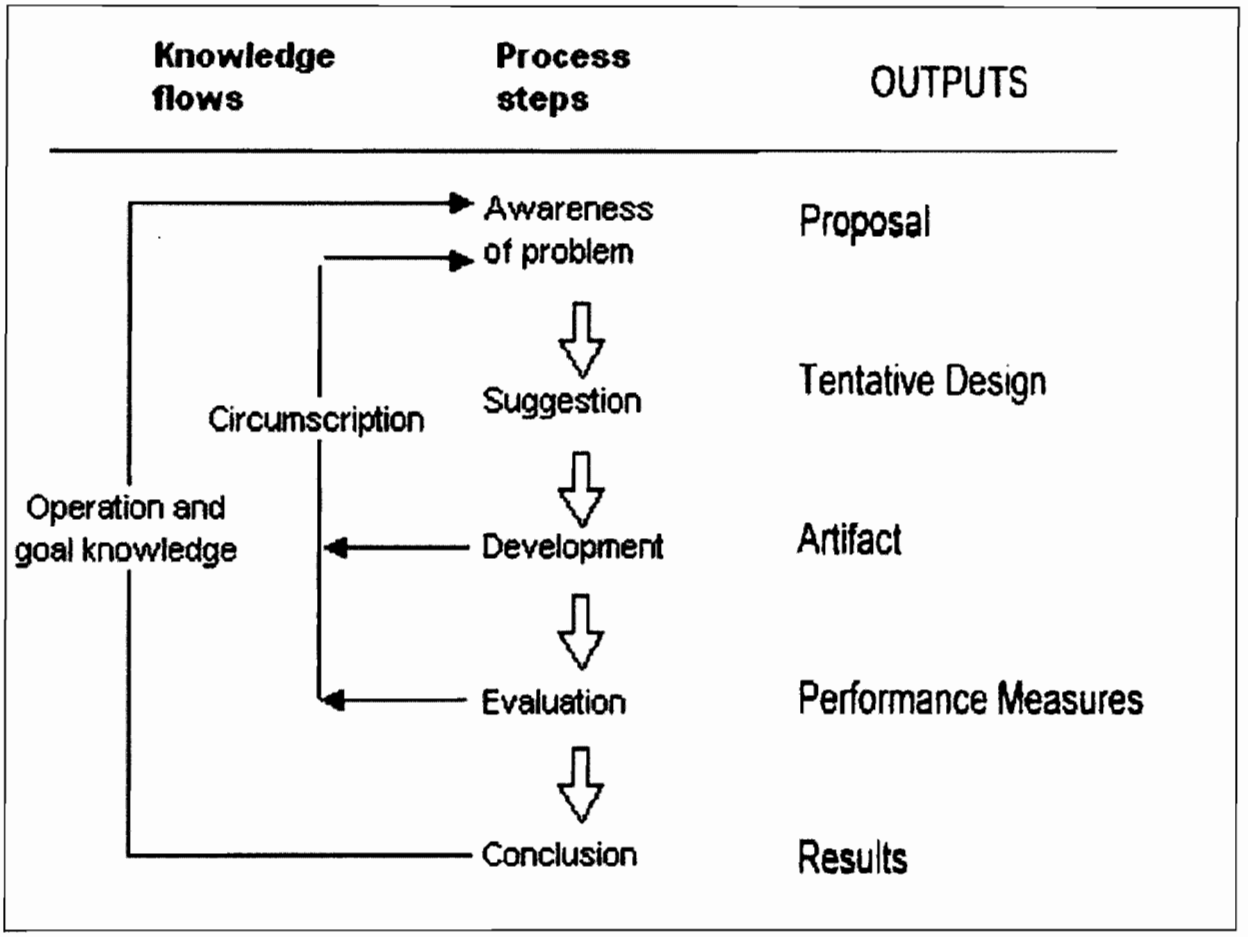

Figure 3.1: General Methodology of Design Research (Vaishnavi & Kuechler, 2004)

As it have been previously mentioned by Vaishnavi and Kuechler (2004), the design research methodology or sometimes called "Improvement Research" contained the major steps: awareness of the problem, suggestion, development, evaluation and conclusion as shown in the Figure 3.1.

3.2 Awareness of the Problem

This kind of problem is common. It is definitely true that human beings are prone to forgetfulness particularly in this modern era were people are made busy by their daily life duties. Imagine a senior manager of a big company forgetting to shut down his/her PC, think about the risk of exposing danger to sensitive data. This common type of mistake can be overcome by equipping the manager with technology that can let manager shut down the PC from a distance.

3.2.1 Interview

I have been met the following:

1. Abdul hakim hafid abdul ghani.
2. Zainol rashed ahmad.
3. Nurul suamira bt md zurri (lab assistance).
4. Nurul ashikin bt nor azian (lab assistance).
5. Mariam bt johari (lab assistance).
6. Zainasrah bt haron (lab assistance).

And we discussed the main point that related with the labs as:

1. How many labs in the IT faculty?
    - There are 9 labs in the IT faculty.

2. How many PC in each lab?

   - There are 40 pc in each lab.

3. How many lab assistances for each lab?

   - There are 2 labs assistance for each lab.

4. What about the current system for network controlling?

   - The manual one which it is if we want to execute an action we should go the pc and do the action.

After these questions I informed them about my idea and we discussed about it so they like it and they precede me to continue.

3.3 Suggestion

In this paper the problem is being attempted to address. The solution for this problem can be a simple application that can provide a command that will execute many tasks to the one PC or more with the help of using Bluetooth technology.

3.4 Development

The prototype will be developed using Microsoft visual studio .net 2005 environment(c#.net). Visual c#.net a modern language as an object oriented, "portable", secure, provides a convenient method for building user interfaces; it

designed to be simple and easy to implement, and the IDE (integrated development environment) minimize the size of code to be written in other languages.

3.5 Evaluation

The evaluation was conducted to determine the amount of the level of usefulness and ease of use according the TAM model of the system after developed the system; it is tested by using a questionnaire, which it was distributed to a lab assistants in UUM, computer center and any person can use the system, the main independent constructs and factors in TAM are the *perceived usefulness* and *perceived ease of use*, *perceived usefulness* and *perceived ease of use* instruments are generally accepted among the research as tools for evaluating the system and predicting usage, the standard variables of *Perceived usefulness* are:

- Decreased the time needed for work.
- Increase the effectiveness of life.
- Improve works.
- Increase opportunities.
- Increase varieties.
- Increase flexibilities.
- Gain job security.

And the standard variables *Perceived Ease of Use* are:

- Easy to get to do what I want it to do.
- Interaction with a mobile device is clear and understandable.
- Easy to use.
- Easy to learn (does not require a lot of mental work to figure out).

The average score of these variables is the measure. The evaluation and the results can be seen in Chapter 6.

3.6 Summary

This chapter has talked about the methodology that used in this research; where the methodology was grouped depends on to four phases based on the research objectives as:

1. Awareness of the problem phase.
2. Suggestion phase.
3. Development phase.
4. Evaluation phase.

CHAPTER 4

RESULTS

4.1 Introduction:

This chapter explains the design of the system and the system development of the *PC Controller* prototype; at the beginning discuss the system requirements and the design, then testing and findings the results of the test conducted for this study, in the end of this chapter the evaluation of the *PC Controller* developed prototype will cover.

4.2 PC Controller Specification

PC Controller has been built to control multi operation on the networked PCs using the mobile by type the order with exact syntax and send it to the one PC by using Bluetooth, it will be received at *Bluetooth exchange folder* and then the networked PCs will take actions according to that order.

4.2.1 Bluetooth Exchange Folder

Devices that have been granted access to the Bluetooth exchange folder also have access to all subfolders contained within that folder and all files in those subfolders. For the simplicity *PC Controller* use the Bluetooth exchange folder as the source of order to the PC so we just need to send order to that folder and the code will manage these orders.

4.2.2 Lists of Requirements

4.2.2.1 Functional Requirements

Stand on the objectives and the definition of the use cases; the following are the functional requirements for *PC Controller.* The complete list of the system requirements (the functional requirements) is shown in the Table 4.1.

Table 4.1: Functional Requirements

List of Requirement	Description
Requirement1: open Bluetooth	The mobile and the PC should open the Bluetooth in the mobile site and in the PC site
Requirement2: search for the Bluetooth	The mobile should search for Bluetooth device to find the PC Bluetooth
Requirement3: Select the PC Bluetooth	At the mobile Bluetooth search select the PC Bluetooth device
Requirement4: Mobile login	The login dialog will appear at the mobile site then the user should type the
Requirement5: PC login	User should type the passkey in the shown dialog at the PC site same passkey in the mobile passkey dialog
Requirement6: select action	There are 4 actions select one in the mobile site
Requirement7: send action	Send the selected action
Requirement8: receive action	The PC will receive the action and save it in the specific folder
Requirement9: do action	The action will happen when the PC save the action and the networked PCs will take the same action.

4.2.2.2 Software Requirements

PC Controller needs some software products as the following:

1. Operating System: Microsoft Windows XP Professional.
2. Bluetooth software: BlueSoleil is the software of Bluetooth device (APPENDIX E)
3. Microsoft Visual C Sharp 2005 Express Edition, This component is essential to run the C#.NET.
4. Mobile Bluetooth software

4.2.2.3 Hardware Requirements

PC Controller needs some hardware products as the following:

1. Network PCs.
2. Bluetooth device(embedded or USB dongle)
3. Mobile supported by Bluetooth

4.3 System Design

The design of the system includes UML diagrams, and a drawing of the system's framework. The UML diagrams contain use case diagram, sequence diagrams and class diagram.

Rational Rose 2000 and Microsoft Visio are used to draw necessary diagrams that help in the development stage. Use case diagram, as displayed in Figure 4.1 describes the overall interaction between the system and its user.

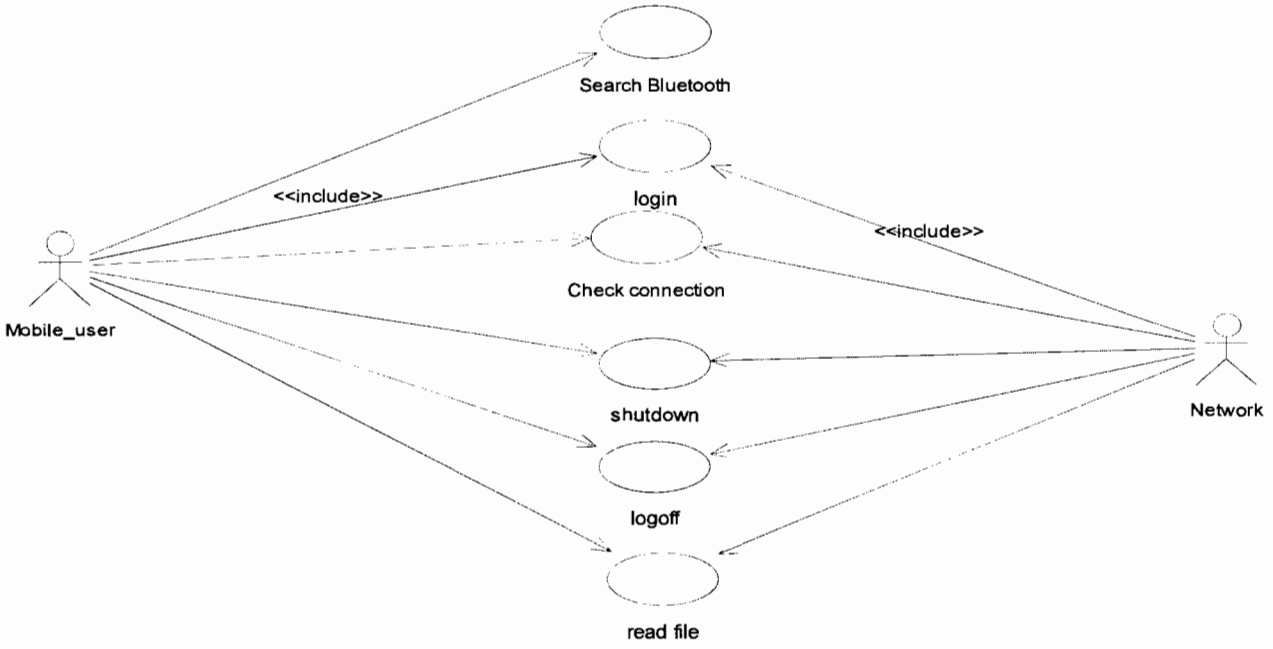

Figure 4.1: Use Case Diagram

Figure 4.1 shows the main use case diagram model the functionality of both network, and mobile-user as actors of the system. The functionality of the mobile-user is the ability to interact with the system by search for network Bluetooth then the user can login to the system and check the network connection then the user can send check network connection to make sure the network its connected with server or not, also the user can send a shutdown to shutdown all the network or logoff the network by send logoff, and also the user can send a file to read it in the network.

4.4 Use Case Specifications

4.4.1 Search Bluetooth

Figure 4.2: Search Bluetooth Diagram

Figure 4.2 shows the case for searching the network Bluetooth from the mobile user.

4.4.1.1 Pre-Condition

- Mobile user Bluetooth is available and the network Bluetooth is available.

4.4.1.2 The Characteristics of Activation

- Execution depends on mobile user demand.

4.4.1.3 Flow of Events

4.4.1.3.1 Basic Flow

- This use case start when the user want search for the PC using the Bluetooth
- So he will turn the Bluetooth on and start searching

4.4.1.3.2 Alternative Flow Cancellation

Cancel the operation.

4.4.1.3.3 Exceptional Flow

-None Exception

4.4.1.4 Post-Conditions

The user can manage this operation

4.4.2 Login

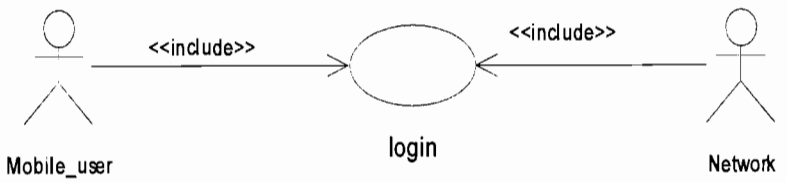

Figure 4.3: Login Diagram

Figure 4.3 shows the case for login from the network site and from network site.

4.4.2.1 Pre-Condition

- The passkey entered in the mobile must reenter the same in the network.

4.4.2.2 The Characteristics of Activation

- Execution depends on mobile user and network demand.

4.4.2.3 Flow of Events

4.4.2.3.1 Basic Flow

- This case starts when the user types the passkey.
- Then in the user should retype the same passkey in the network.

4.4.2.3.2 Alternative Flow Cancellation

Cancel the operation.

4.4.2.3.3 Exceptional Flow

-None

4.4.2.4 Post-Conditions

The user can access the network Bluetooth

4.4.3 Check Connection

Figure 4.4: Check Connection Diagram

Figure 4.4 shows the case for check connection action the user send and the network receive.

4.4.3.1 Pre-Condition

- Mobile user must send the action check to the network.

4.4.3.2 The Characteristics of Activation

- Execution depends on mobile user and network demand.

4.4.3.3 Flow of Events

4.4.3.3.1 Basic Flow

- This use case start when the user send Bluetooth message to check the network connection

- Then the network will receive the message and do the action to check the connection

4.4.3.3.2 Alternative Flow Cancellation

Cancel the operation.

4.4.3.3.3 Exceptional Flow

-None

4.4.3.4 Post-Conditions

The user can know if the network connected or not.

4.4.4 Shutdown

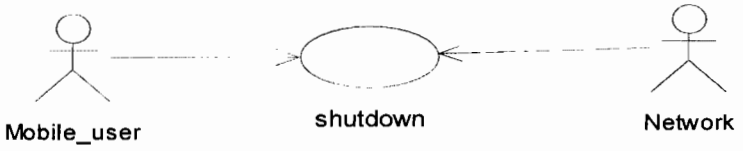

Figure 4.5: Shutdown Diagram

Figure 4.5 shows the case for shutdown all the networked PCs.

4.4.4.1 Pre-Condition

- Mobile user must send the action shutdown to the network.

4.4.4.2 The Characteristics of Activation

- Execution depends on mobile user and network demand.

4.4.4.3 Flow of Events

4.4.4.3.1 Basic Flow

- This use case starts when the user sends Bluetooth message to shutdown the networked PCs.
- Then the network will receive the message and do the action to shutdown the networked PCs.

4.4.4.3.2 Alternative Flow Cancellation

Cancel the operation.

4.4.4.3.3 Exceptional Flow

-None

4.4.4.4 Post-Conditions

The user can shutdown all the networked PCs.

4.4.5 Logoff

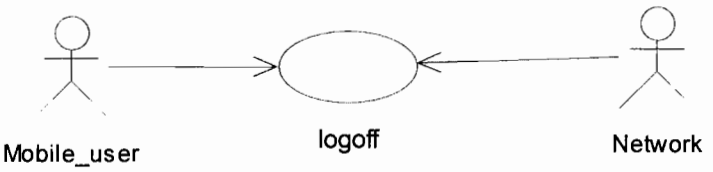

Figure 4.6: Logoff Diagram

Figure 4.5 shows the case for logoff all the networked PCs.

4.4.5.1 Pre-Condition

- Mobile user must send the action logoff to the network.

4.4.5.2 The Characteristics of Activation

- Execution depends on mobile user and network demand.

4.4.5.3 Flow of Events

4.4.5.3.1 Basic Flow

- This use case starts when the user sends Bluetooth message to logoff the networked PCs.

- Then the network will receive the message and do the action to logoff the networked PCs.

4.4.5.3.2 Alternative Flow Cancellation

Cancel the operation.

4.4.5.3.3 Exceptional Flow

-None

4.4.5.4 Post-Conditions

The users can logoff all the networked PCs.

4.4.6 Read File

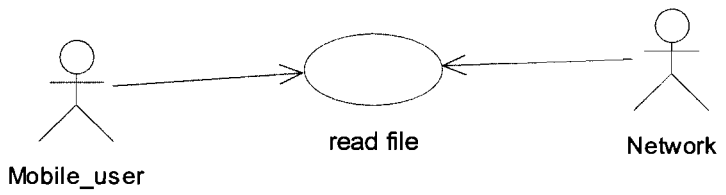

Figure 4.7: Read File Diagram

Figure 4.5 shows the case for read a file has been sent to all the networked PCs.

4.4.6.1 Pre-Condition

- Mobile user must send the action read file to the network.

4.4.6.2 The Characteristics of Activation

- Execution depends on mobile user and network demand.

4.4.6.3 Flow of Events

4.4.6.3.1 Basic Flow

- This use case starts when the user sends Bluetooth message to read file in the all networked PCs.
- Then the network will receive the message and do the action to read file in the all networked PCs.

4.4.6.3.2 Alternative Flow Cancellation

Cancel the operation.

4.4.6.3.3 Exceptional Flow

-None

4.4.6.4 Post-Conditions

The user able to send a file and appear it in the all networked PCs.

4.5 Sequence Diagrams:

4.5.1 Search Bluetooth

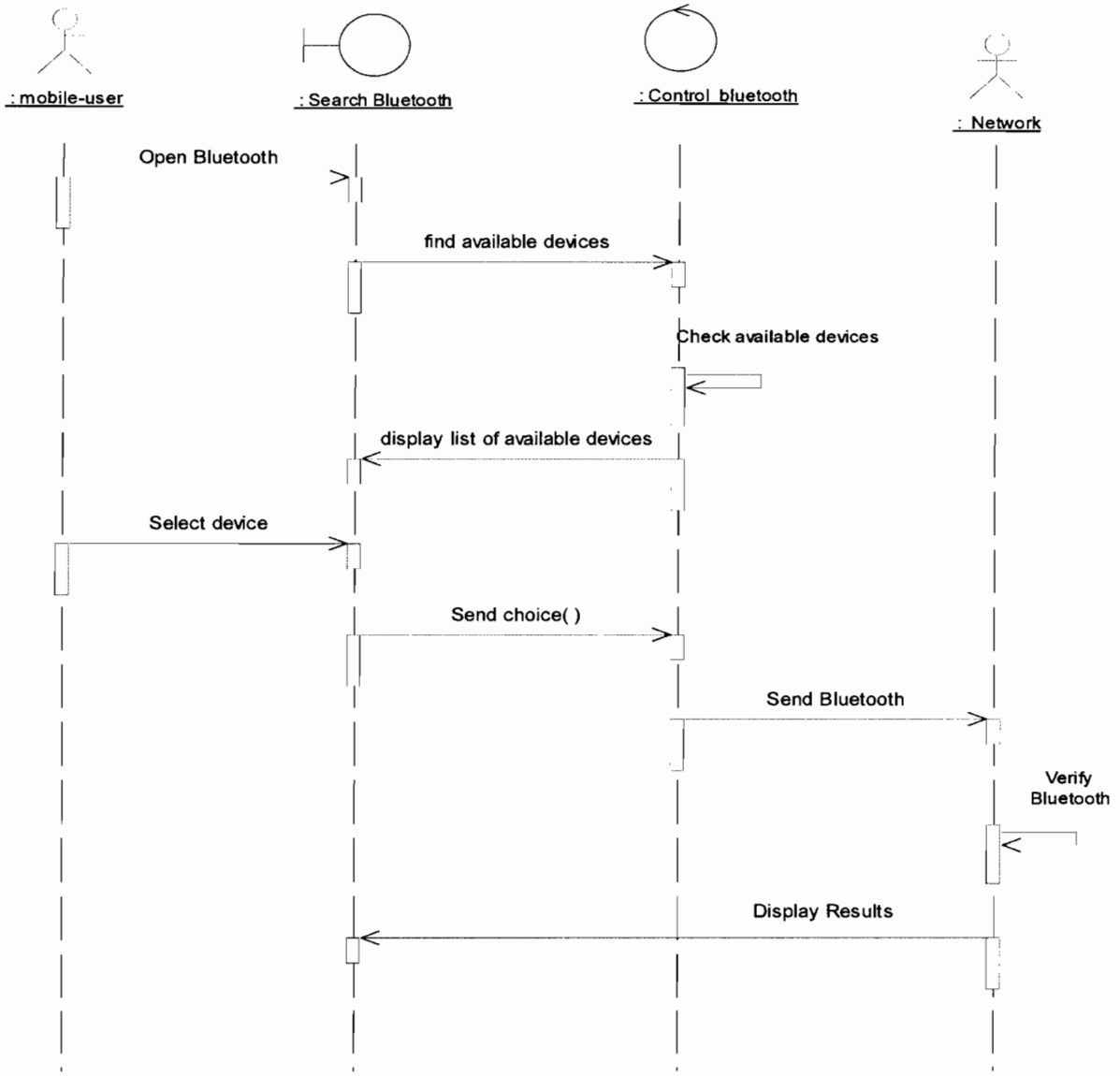

Figure 4.8: Search Bluetooth Sequence Diagram

Figure 4.8 shows the search Bluetooth sequence diagram; in this diagram the user should open the Bluetooth in the mobile then the user can search for the available Bluetooth and can also chose the network Bluetooth.

4.5.2 Login

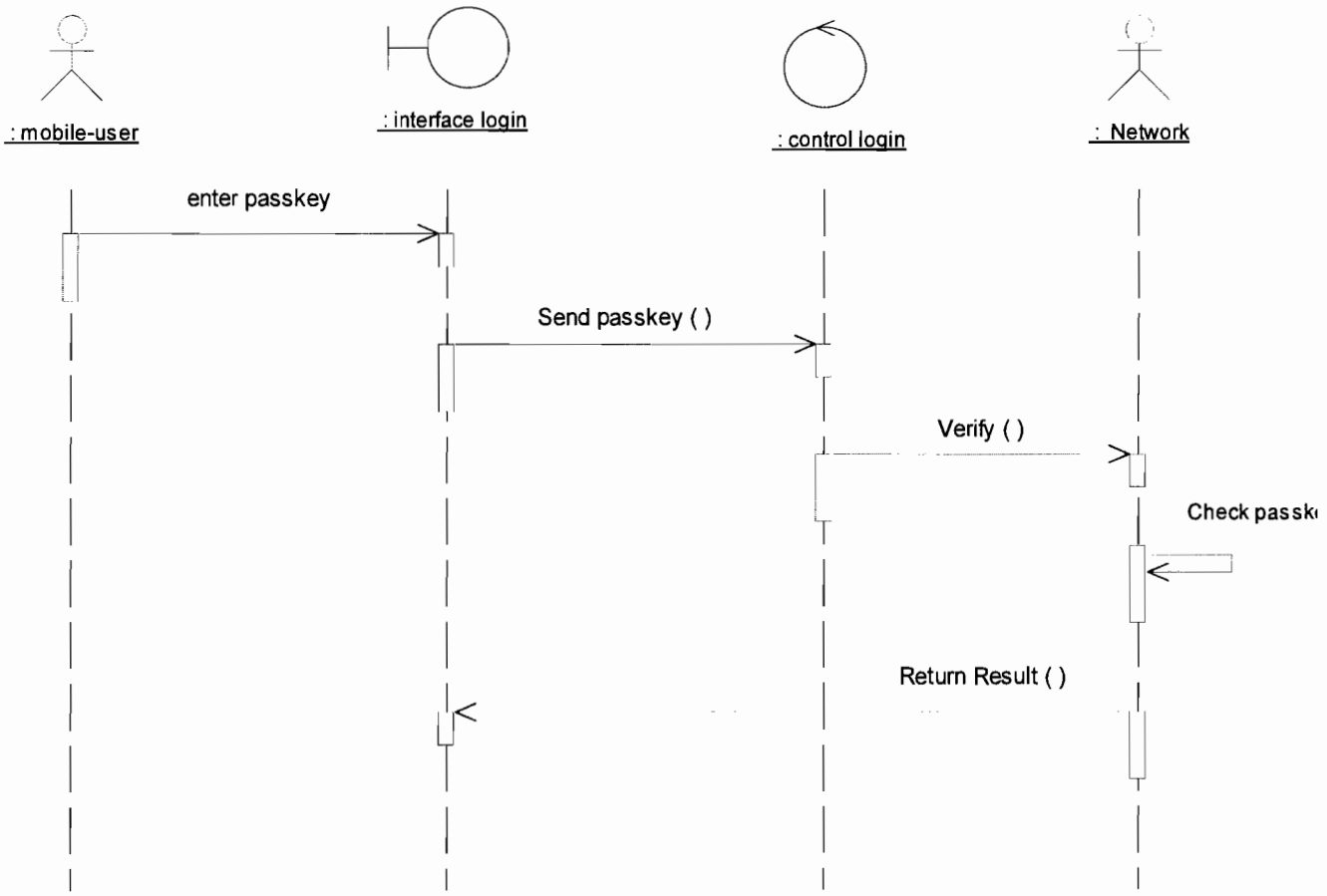

Figure 4.9: Login Sequence Diagram

Figure 4.9 shows the login sequence diagram; in this diagram when the user send invitation to the network to accept the connection order. The user must enter the passkey to be connected with the network and from the network the user should type the same passkey in the network.

4.5.3 Check Connection

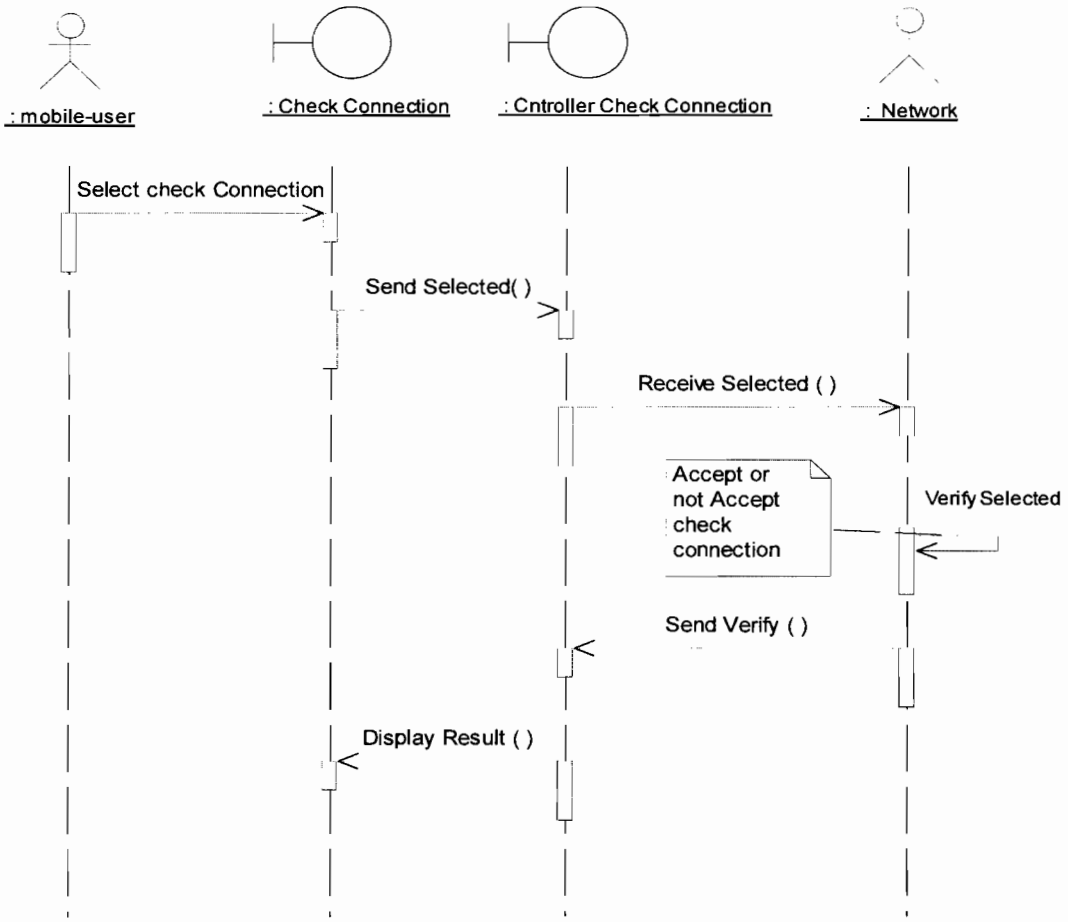

Figure 4.10: Check Connection Sequence Diagram

Figure 4.10 shows the check connection sequence diagram; in this diagram the user can select the check connection choice, and send it to check the network connection, then the network PCs will check the connection.

4.5.4 Shutdown

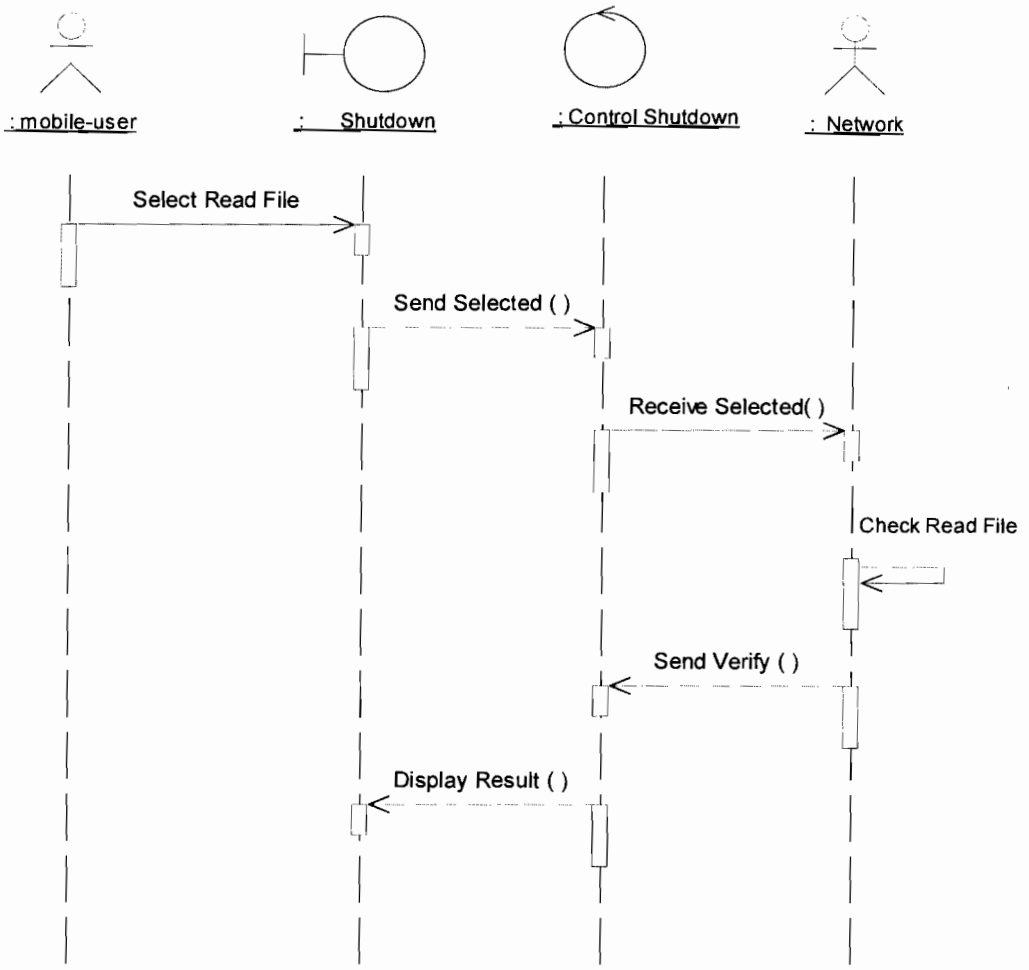

Figure 4.11: Shutdown Sequence Diagram

Figure 4.11 shows the shutdown sequence diagram; in this diagram the user can select the shutdown choice, and send it to shutdown all the networked PCs, then the networked PCs will shutdown.

4.5.5 Logoff

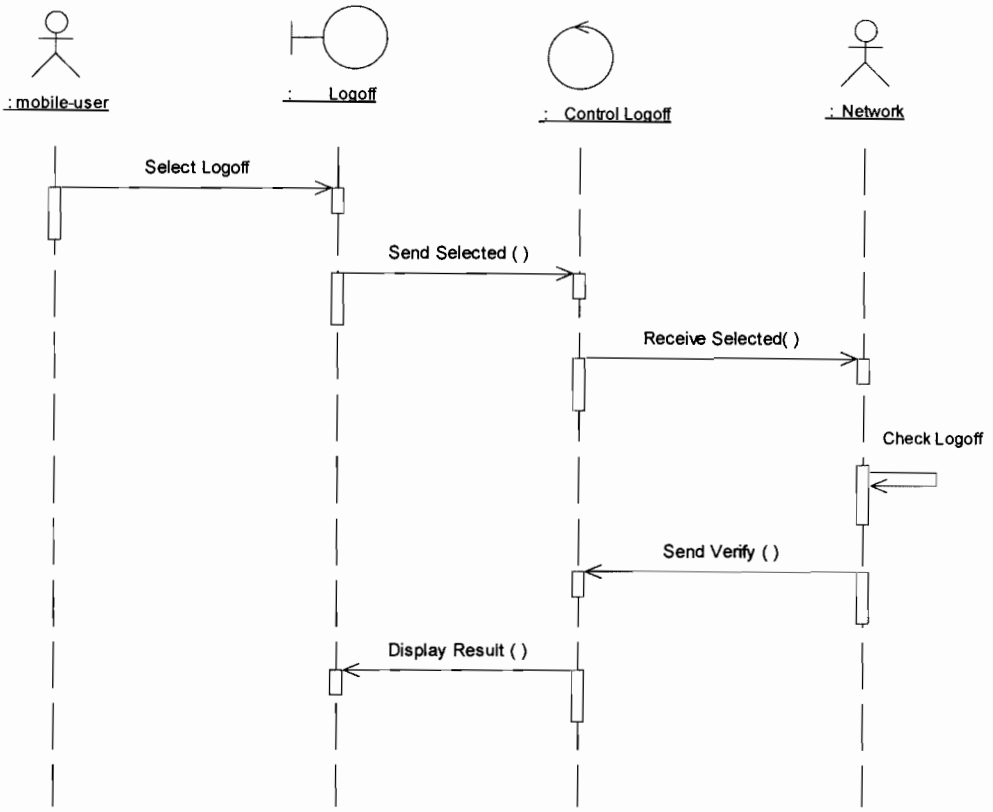

Figure 4.12: Logoff Sequence Diagram

Figure 4.12 shows the logoff sequence diagram; in this diagram the user can select the logoff choice, and send it to logoff all the networked PCs, then the networked PCs will logoff.

4.5.6 Read File

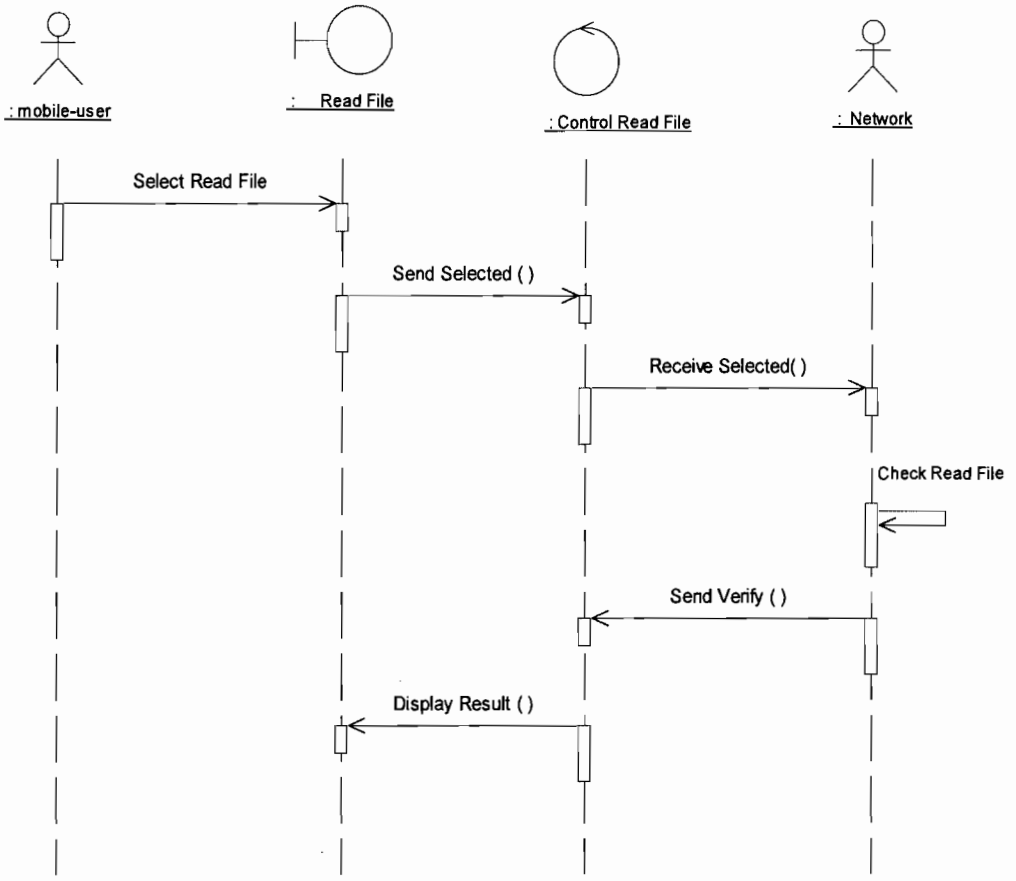

Figure 4.13: Read File Sequence Diagram

Figure 4.12 shows the read file sequence diagram; in this diagram the user can select the logoff choice, and send it to read file in the all networked PCs, then the network will receive the file and do the action to read this file in the all networked PCs.

4.6 System Framework

There are three main entities in the proposed framework for *PC Controller* System.

4.6.1 PC Controller Users

The system users (lab assistant, Network Administrator) initiate communication with the system server requesting access to the Bluetooth. Therefore the user should open the Bluetooth from his/her mobile then search for server Bluetooth then type the passkey in mobile and retype the same passkey in the server to be authenticated to control the network.

4.6.2 Bluetooth Serves

The Bluetooth serves determines whether to grant the service or not, based on the passkey it is match with mobile passkey or not.

4.6.3 System Infrastructure

The main idea of the system is controlling the network using the Bluetooth by send file to the PC to execute some action. Now I will elaborate the multi action that this system provide for example the first thing must happen is

searching for the pc through the mobile Bluetooth then after founding the pc the process will start by sending the file to execute the action as the test the internet connection or log off for the pc or shut it down or display some text on the PC.

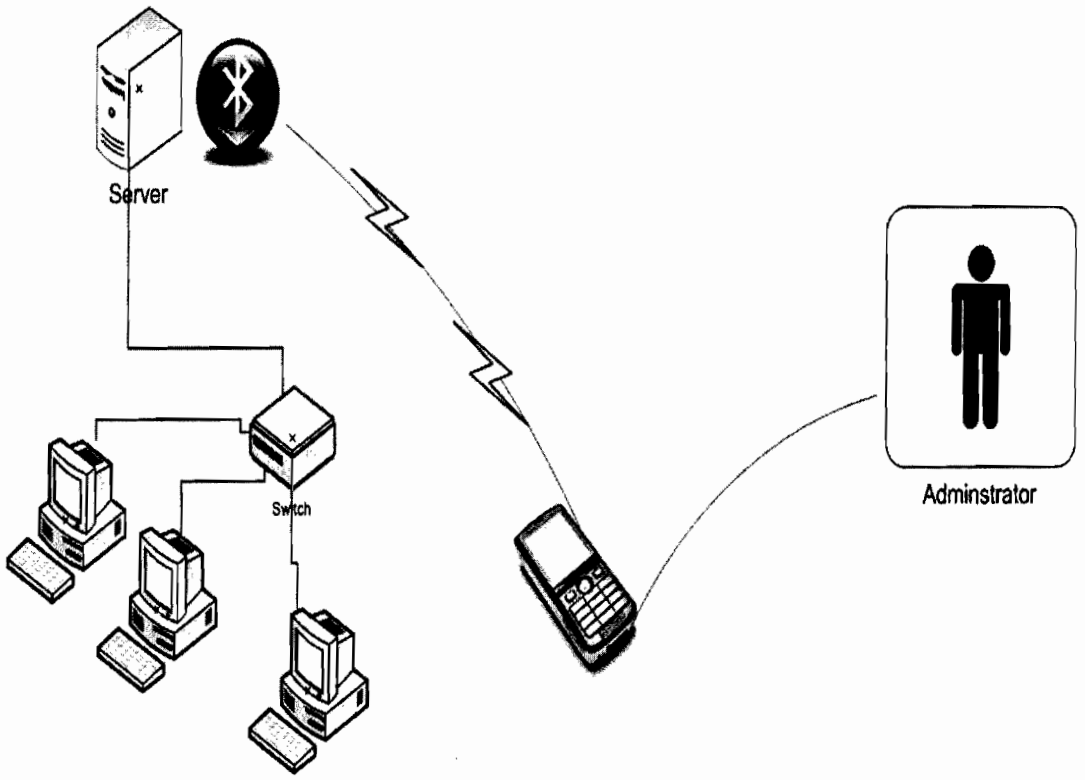

Figure 4.14: System Framework

4.7 PC Controller System Implementation

4.7.1 The Main Screen For Server

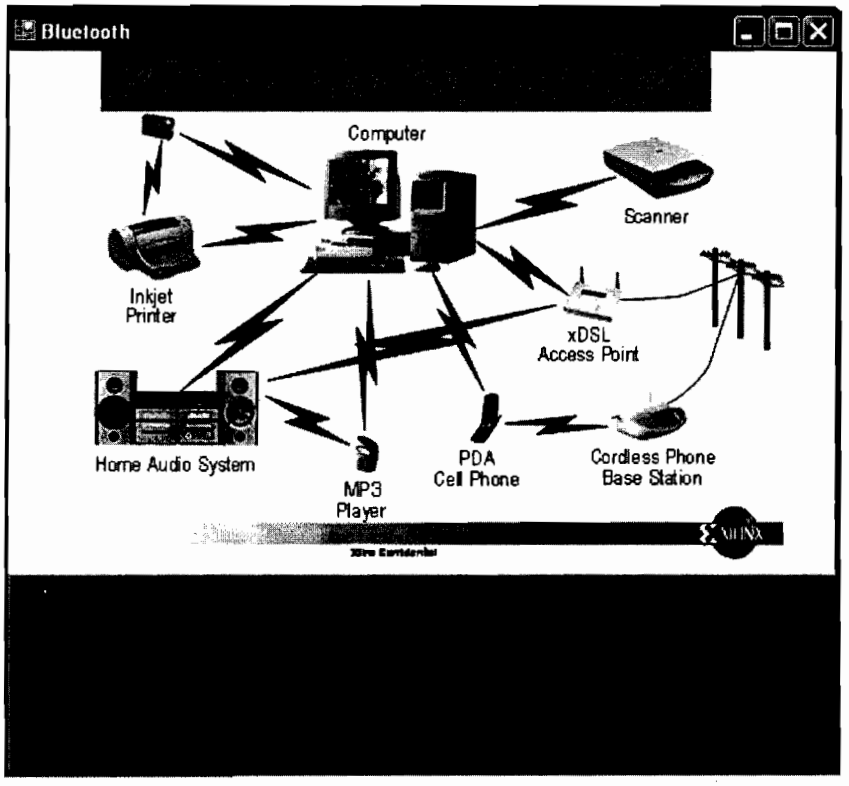

Figure 4.15: System Main Screen for Server

Figure 4.15 shows the system main screen for server, this screen shown when the user open the *PC controller* in the server to be able to do the actions come from the mobile.

4.7.2 The Main Screen For Client

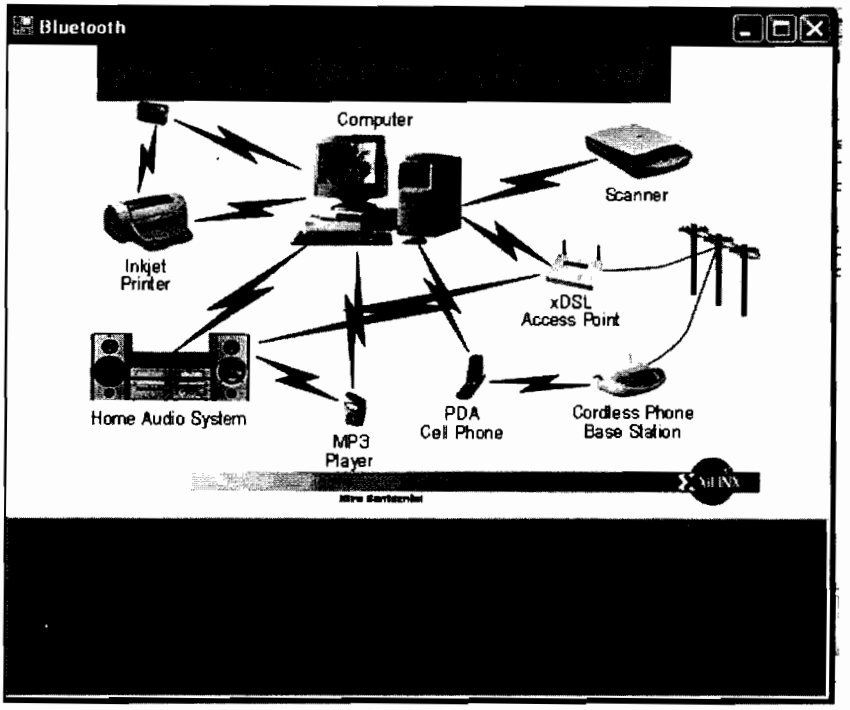

Figure 4.16: System Main Screen for Client

Figure 4.16 shows the system main screen for client, this screen shown when the user open the *PC controller* in the client to be able to do the actions come from server.

4.7.3 Mobile Search

The mobile user should search for available Bluetooth to find Bluetooth devices to make a connection with the server; in this situation the Bluetooth server should be turn on, so the other devices can find it to make a connection.

(a) (b)

(c)

Figure 4.17: Search Bluetooth in the Mobile

The mobile user should open the Bluetooth from the mobile as it shown in the Figure 4.17 (a). Then the mobile user should search for available Bluetooth Figure 4.17 (b). Then the mobile Bluetooth should find a server Bluetooth and available Bluetooth the mobile user should select the server Bluetooth as it shown in the Figure 4.17 (c).

4.7.4 Mobile Login

Figure 4.18: Mobile Login

When the mobile user sends an invitation to the Bluetooth server, then the server will send a request to the mobile to enter the passkey as it shown in the Figure 4.18.

4.7.5 Server Login

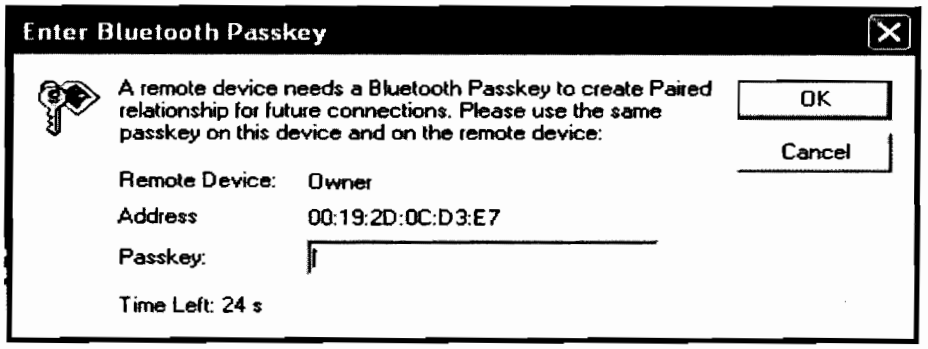

Figure 4.19: Server Login

After the entered the passkey in the mobile the server will show a dialog box to the user to use the same passkey as it shown in the Figure 4.19.

4.7.6 The Actions

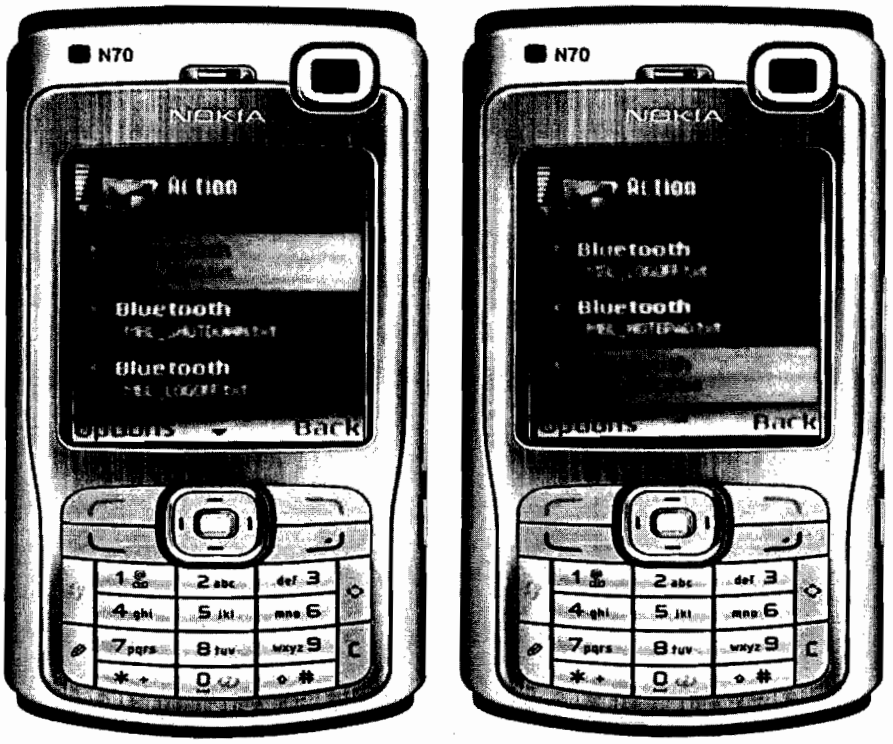

Figure 4.20: Mobile Actions.

The mobile user have some actions (shutdown, logoff, read file and check network connection) can do it from the mobile to control the network and the actions will appear as it shown in the Figure 4.20.

4.7.6.1 Check the Network Connection

Figure 4.21: Check Network Connection

As a test the lab assistant can know if the computer connected to the network or not by send a file its name *MBL_CHECK.TXT* as it shown in the Figure 4.21, as a result the server and the client will show a message box tell if the computer connected or not. The server will show the message as it shown in Figure 4.22.

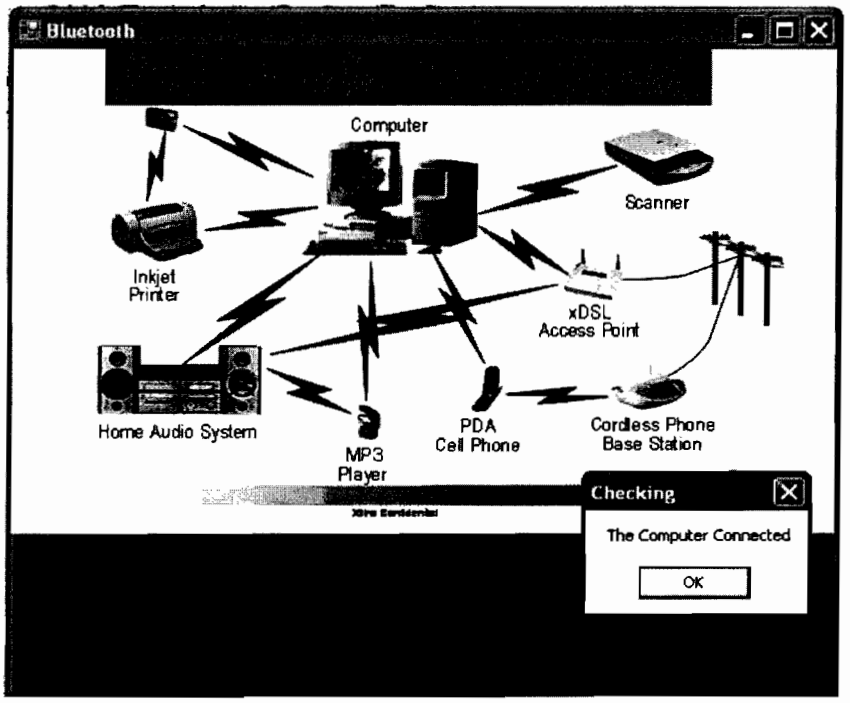

Figure 4.22: Server Check Network Connection

As it shown in the Figure 4.22 at the first the server should create a file in the specific path and shows the message "(MBL_CHECK.txt) has been created in the directory", and then the server will shows message box to tell the user "The connection message showed", at the last when the action done the server will delete the file from the directory when the user read the message and click ok button, to ensure there are not conflict will happen if the user send the message again. At the same time the action will appear at the client site as it shown in the Figure 4.23.

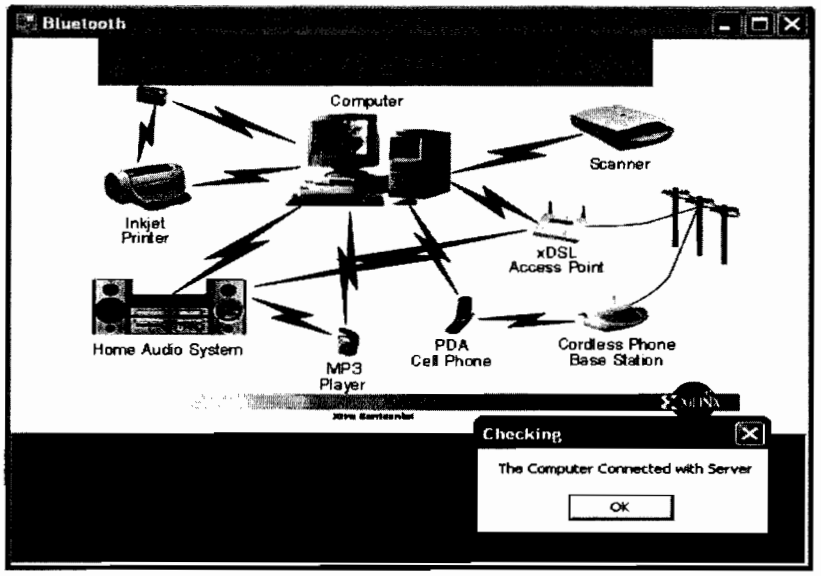

Figure 4.23: Client Check Network Connection

The client does not allow to creating the file in the directory and delete it only shows the connection message showed.

4.7.6.2 Read Text Document and Views It

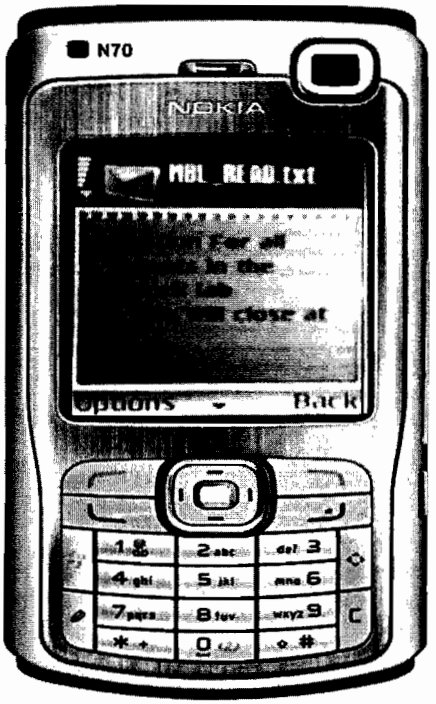

Figure 4.24: Write Text Message

If the lab assistant want to send an attention to the students in the lab to tell them the lab will close at the specific time the lab assistant will write the text message and name it MBL_READ.TXT as it shown in the Figure 4.24 and then send it by Bluetooth, as a result its will be opened and viewing on the terminal. At the server site the message will appear as it shown in the Figure 4.25.

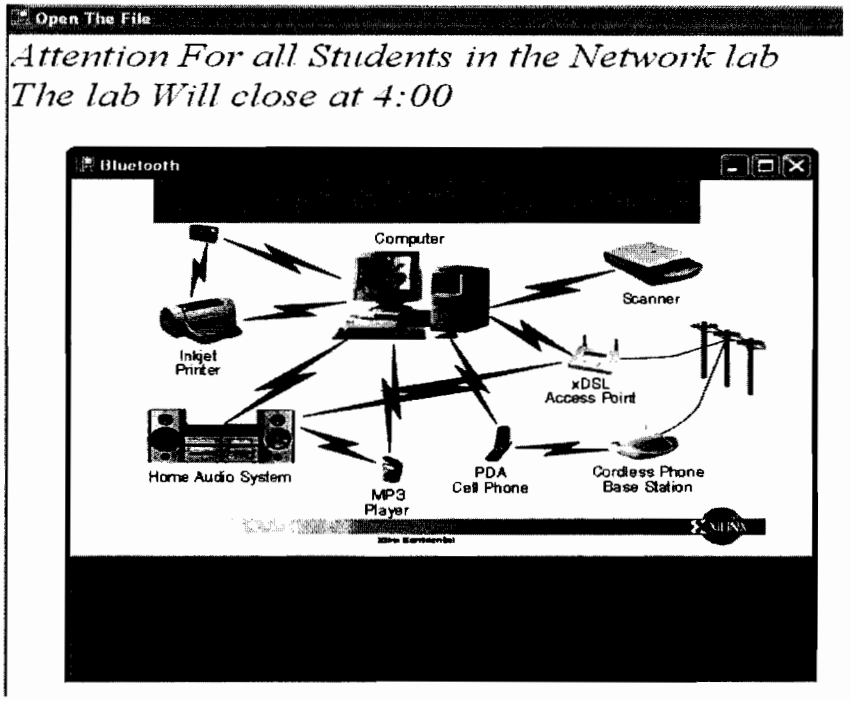

Figure 4.25: Read Text File and Show It in the Server

As it shown in the Figure 4.25 at the first the server should create a file in the specific path and shows the message (MBL_READ.TXT) has been created in the directory, and then the server will shows message to tell the user "the file opened", at the last when the action done the server will delete the file from the directory, to ensure there are not conflict will happen if the user send the

message again. At the same time the action will appear at the client site as it shown in the Figure 4.26.

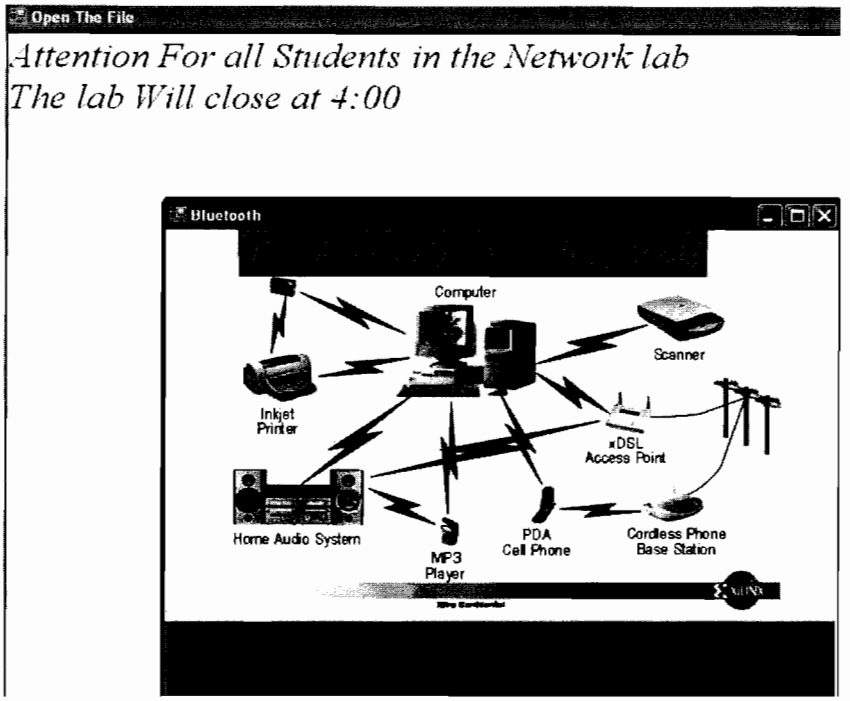

Figure 4.26: Read Text File and Show It in the Client

The client does not allow to creating the file in the directory and delete it only shows the file opened.

4.7.6.3 Logoff the Computer

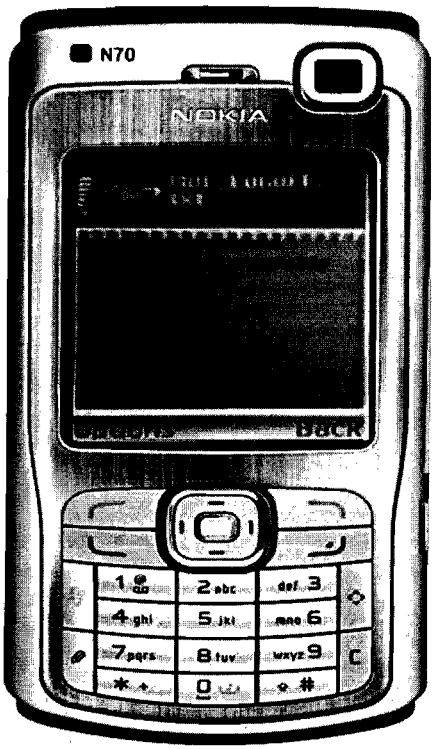

Figure 4.27: Logoff File.

Computer Logoff action provides functionality to log off the networked computers, at a time that the lab assistant chooses it by send message or file its name MBL_LogOff.TXT as it shown in the Figure 4.27. Functionality of the log off modes has been provided by a WindowsController.cs, and the need this option some times to log off the computer from far destination.

Warning: This option will force termination of all running processes; any unsaved data will be lost; guaranteeing a log off. This option has been designed to close applications which produce a prompt upon closing.

4.7.6.4 Shutdown the Computer

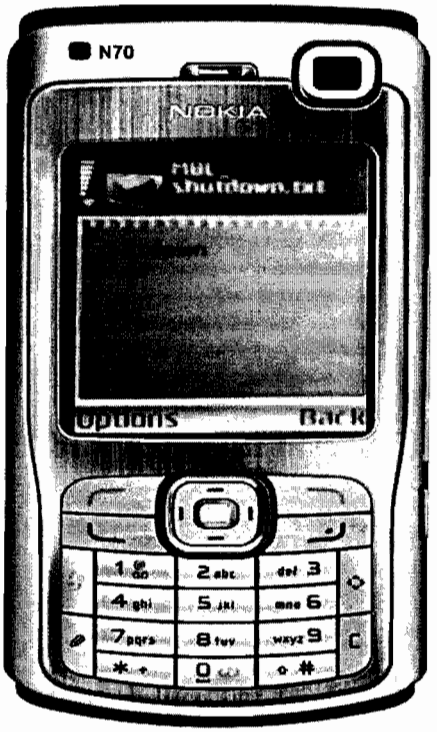

Figure 4.28: Shutdown File.

Computer Shutdown action provides functionality to Shut down the networked computers, at a time that the lab assistant chooses it by send message or file its name Shutdown.txt as it shown in the Figure 4.28.

Functionality of the Shut down modes has been provided by a *WindowsController.cs,* and the need this option some times to shutdown the computer from far destination.

Warning: This option will force termination of all running processes; any unsaved data will be lost; guaranteeing a log off. This option has been designed to close applications which produce a prompt upon closing.

4.8 Summary

In this chapter, the design and implementation of the several steps for building the system is discussed and tested. A PC controller for any network was developed. The result of running the system showed that objective of the study is done successfully. The output of this chapter is elaborating the execution of this the study.

CHAPTER 5

RESULT DISSCUSION

5.1 Introduction

The main aim of this chapter is to discuss the evaluation of the *PC controller* developed prototype. A usability test is one of the most fundamental methods in usability evaluation, because real test users are asked to use the product. The moderator of the test gives predetermined test tasks one at a time to the test user, who in turn performs the tasks with the user interface (Nielson, 1993). The users are usually asked to think while doing the test tasks. Interviews are also often used in order to gain more insight into the user's actions with the interface.

5.2 Evaluation Techniques

The evaluation was performed after the system has been developed to determine the level of usefulness and operability of the system; it is tested through a questionnaire which was distributed to the public. The sample size was 30 respondents (Nielsen, 2006); each participant was given a brief description of the functionality of PCCA afterwards, they were allowed to practice and explore the prototype, finally were given a set of prepared questionnaire to obtain their

perceptions. The aim was to see the level of satisfaction and perception of the developed prototype ease of use and operability of the prototype system.

5.3 Evaluation Questionnaire

The questionnaire questions were prepared and adopted from different standard questionnaire (Davis, 1989), it consisted of two main sections, firstly general information which intended to gather demographic data about the sample and its distribution. The second part included questions about the perceptions of the participant regarding different dimensions of usability (Usefulness, Ease of Use) the questions were close ended and scaled from "Strongly disagree" to "strongly agree". A form of the questionnaire where attached in Appendix A.

5.4 Data Analysis

A 35 respondent was distributed where 5 considered as untrusted answers and the SPSS conceders them as missing and delete them. Table 5.1 will show the reliability statistics for this study and it is = 84.4 % which it is mean acceptable. According to Hair et al., (2006) when the Cronbach alpha is more than 60% the study is acceptable.

Table 5.1: Reliability Statistics

Cronbach's Alpha	N of Items
.844	14

Table 5.2 shows the mean and the STD deviation and the number of the respondent for the system aspect questions.

Table 5.2: Item Statistics

	Mean	Std. Deviation	N
systemaspects1	3.83	.699	30
systemaspects2	4.00	.695	30
systemaspects3	4.10	.759	30
systemaspects4	3.93	.740	30
systemaspects5	4.00	.743	30
systemaspects6	3.97	.890	30
systemaspects7	4.03	.765	30
systemaspects8	4.03	.765	30
systemaspects9	4.27	.785	30
systemaspects10	4.17	.747	30
systemaspects11	4.23	.679	30
systemaspects12	3.93	.740	30
systemaspects13	4.13	.629	30
systemaspects14	4.37	.850	30

Table 5.3 will show the percentage of scale mean in item deleted and the scale variance item deleted and the corrected item total correlation and the Cronbach's alpha item deleted for the system aspect questions.

Table 5.3: Item Total Statistics

	Scale Mean if Item Deleted	Scale Variance if Item Deleted	Corrected Item-Total Correlation	Cronbach's Alpha if Item Deleted
systemaspects1	53.17	32.971	.385	.840
systemaspects2	53.00	30.414	.738	.819
systemaspects3	52.90	31.266	.555	.829
systemaspects4	53.07	31.168	.586	.827
systemaspects5	53.00	31.931	.485	.834
systemaspects6	53.03	31.689	.406	.840
systemaspects7	52.97	32.240	.429	.837
systemaspects8	52.97	32.102	.446	.836
systemaspects9	52.73	30.754	.595	.826
systemaspects10	52.83	30.282	.695	.820
systemaspects11	52.77	32.047	.526	.832
systemaspects12	53.07	33.237	.325	.843
systemaspects13	52.87	35.499	.088	.854
systemaspects14	52.63	30.930	.518	.832

Table 5.4 will describe the final percentage for the mean and the variance and STD deviation and the number of the respondent for the system aspect questions.

Table 5.4: Scale Statistics

Mean	Variance	Std. Deviation	N of Items
57.00	36.552	6.046	14

Now we will elaborate the general question part as bellow and we will start with the gender, Table 5.5 will present the frequency and the percentages for the gender in case male or female in chart 5.1 you can see that the high percentage was to the male respondent it was 57% where the female was 43%.

Table 5.5: Gender

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	male	17	48.6	56.7	56.7
	female	13	37.1	43.3	100.0
	Total	30	85.7	100.0	
Missing	System	5	14.3		
Total		35	100.0		

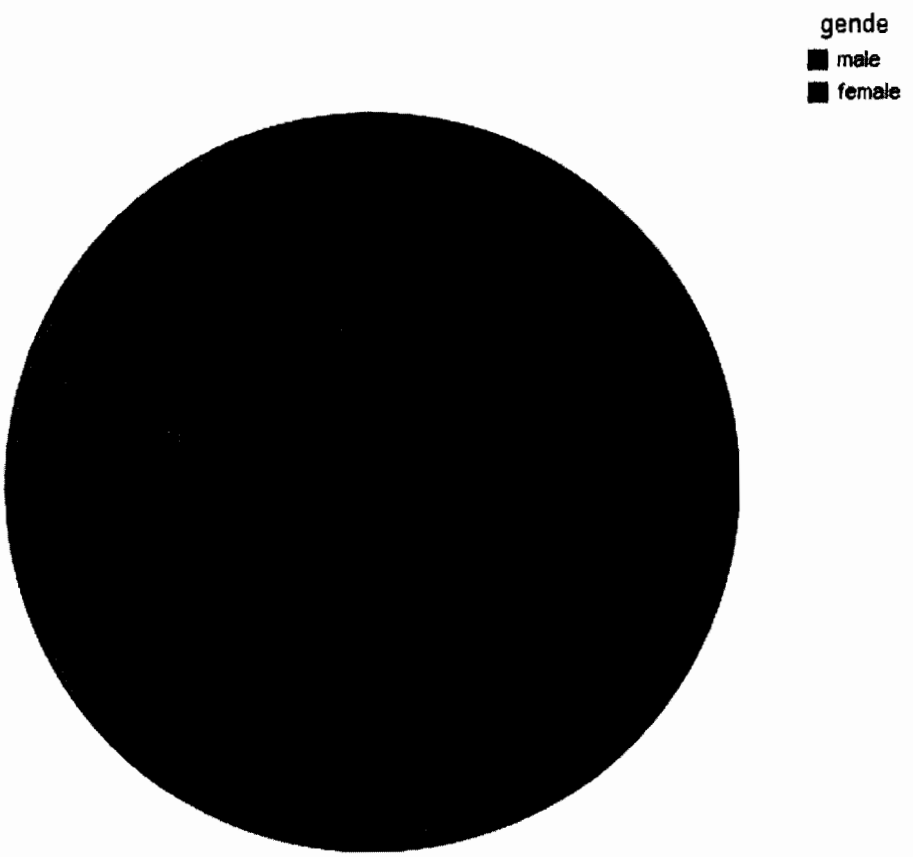

Chart 5.1: Gender

Table 5.6 shows the frequency and percentages of the age range and chart 5.2 show that the highly percentage for the respondent was to the people who there age is between 21-30 it was 77%.

Table 5.6: Age

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	<20 year	2	5.7	6.7	6.7
	21-30 years	23	65.7	76.7	83.3
	31-40 years	4	11.4	13.3	96.7
	>40 years	1	2.9	3.3	100.0
	Total	30	85.7	100.0	
Missing	System	5	14.3		
Total		35	100.0		

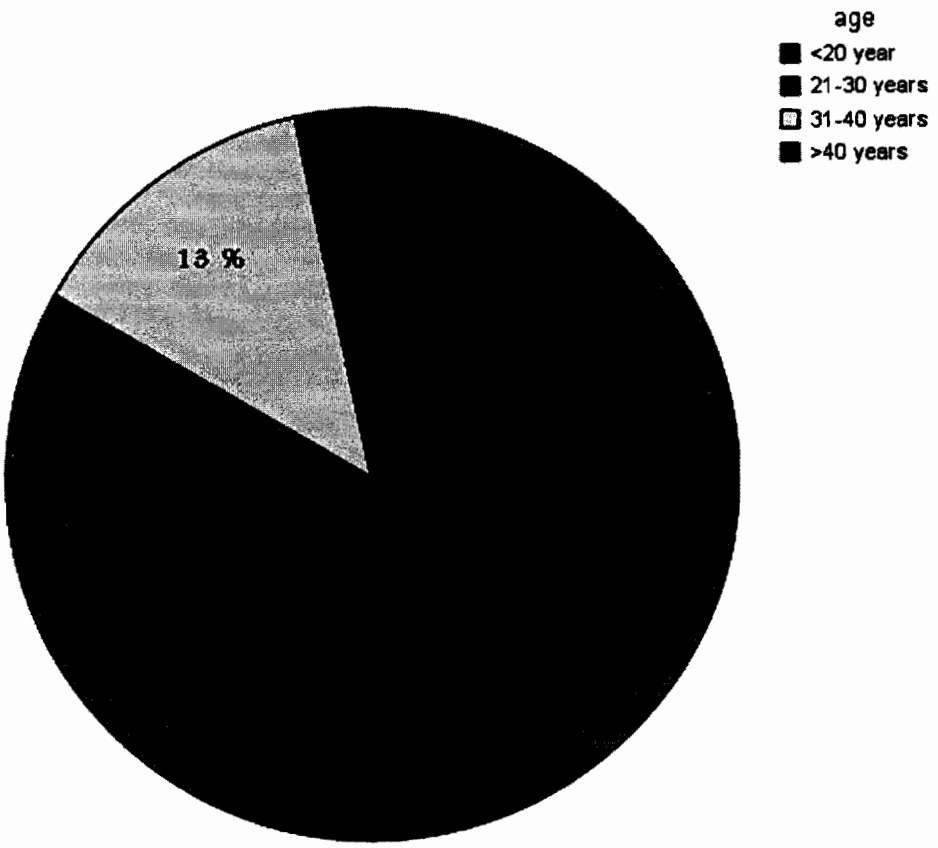

Chart 5.2: Age

Table 5.7 is elaborating the frequency and the percentage for the education, chart 5.3 show the percentage for the respondent regarding to the education so the high percent was to the people who there pack ground is master it was 63% see chart 5.3 .

Table 5.7: Education

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	diploma	4	11.4	13.3	13.3
	degree	6	17.1	20.0	33.3
	master	19	54.3	63.3	96.7
	phD	1	2.9	3.3	100.0
	Total	30	85.7	100.0	
Missing	System	5	14.3		
Total		35	100.0		

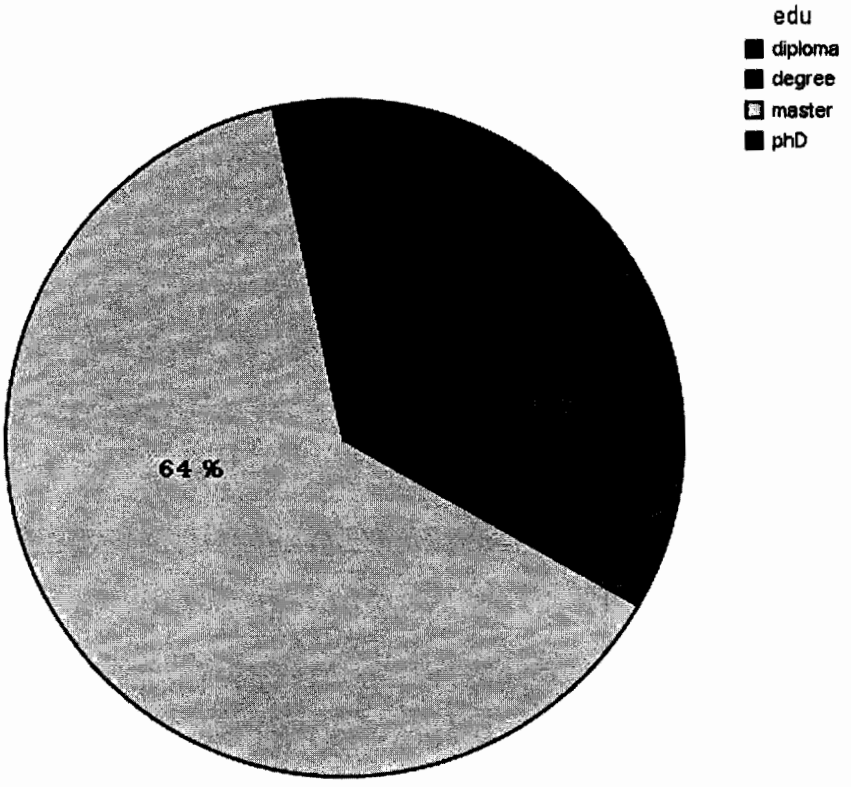

Chart 5.3: Education

Table 5.8 shows the frequency and the percentage for the period of using mobile a 33% was to the period above 3 years and the none see chart 5.4.

Table 5.8: Period

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	1-2 year	4	11.4	13.3	13.3
	2-3 year	6	17.1	20.0	33.3
	> 3 year	10	28.6	33.3	66.7
	none	10	28.6	33.3	100.0
	Total	30	85.7	100.0	
Missing	System	5	14.3		
Total		35	100.0		

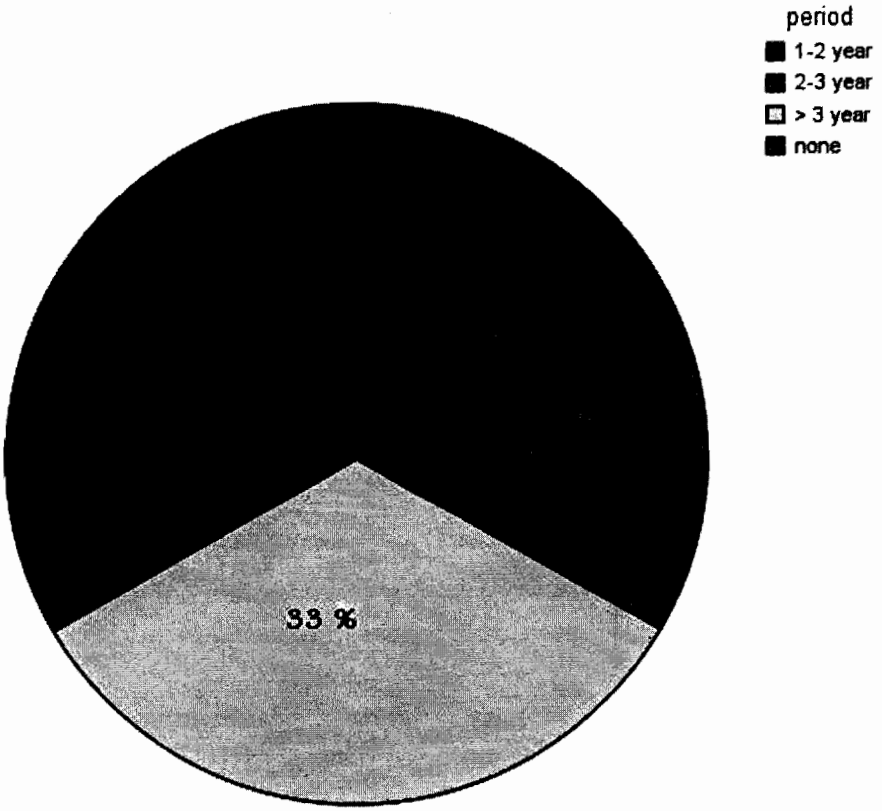

Chart 5.4: Period

Table 5.9 shows the frequency and the percentage for the mobile brand most percentage was to the nokia users it was 67% see chart 5.5.

Table 5.9: Brand

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	motorolla	4	11.4	13.3	13.3
	nokia	20	57.1	66.7	80.0
	sony	2	5.7	6.7	86.7
	samsong	2	5.7	6.7	93.3
	semens	2	5.7	6.7	100.0
	Total	30	85.7	100.0	
Missing	System	5	14.3		
Total		35	100.0		

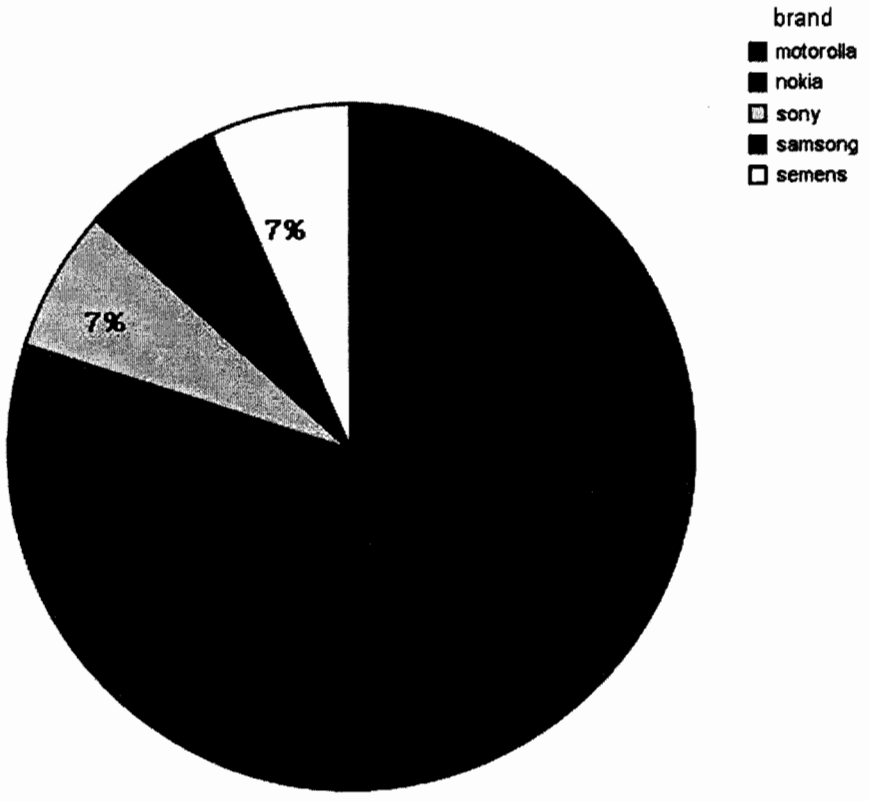

Chart 5.5: Brand

Table 5.10 well describe the system aspect answers and elaborate the number of the respondent and mean and the STD deviation for each question, now we will elaborate the two dimensions which they are usefulness and ease of use according to TAM, in the following table I present the first dimension which it is usefulness from systemaspect1 to systemaspect8 in appendix A it is from question 6 to question 13 so the mean for this dimension is 4.01 which present high percent according TAM. Regarding the second dimension is ease of use in the following table it is from systemaspect9 to systemaspect14 in the appendix A it is the questions from 14 to 19 so the mean for this dimension is 4.18. The mean for the two dimensions is calculated using the SPSS.

Table 5.10: Descriptive Statistics (system aspect)

	N	Minimum	Maximum	Mean	Std. Deviation
systemaspects1	30	3	5	3.83	.699
systemaspects2	30	3	5	4.00	.695
systemaspects3	30	2	5	4.10	.759
systemaspects4	30	2	5	3.93	.740
systemaspects5	30	2	5	4.00	.743
systemaspects6	30	2	5	3.97	.890
systemaspects7	30	3	5	4.03	.765
systemaspects8	30	2	5	4.03	.765
systemaspects9	30	2	5	4.27	.785
systemaspects10	30	2	5	4.17	.747
systemaspects11	30	3	5	4.23	.679
systemaspects12	30	2	5	3.93	.740
systemaspects13	30	3	5	4.13	.629
systemaspects14	30	1	5	4.37	.850
Valid N (listwise)	30				

Chart 5.6 show the system aspect variable (the total of 14 questions) after compute the 14 system aspect questions using a formula in SPSS we will treat the 14 question by one variable called system aspect so will found the total mean and the chart 5.6 show how the mean will be distributed between a range less than 3.50 and to above 4.50

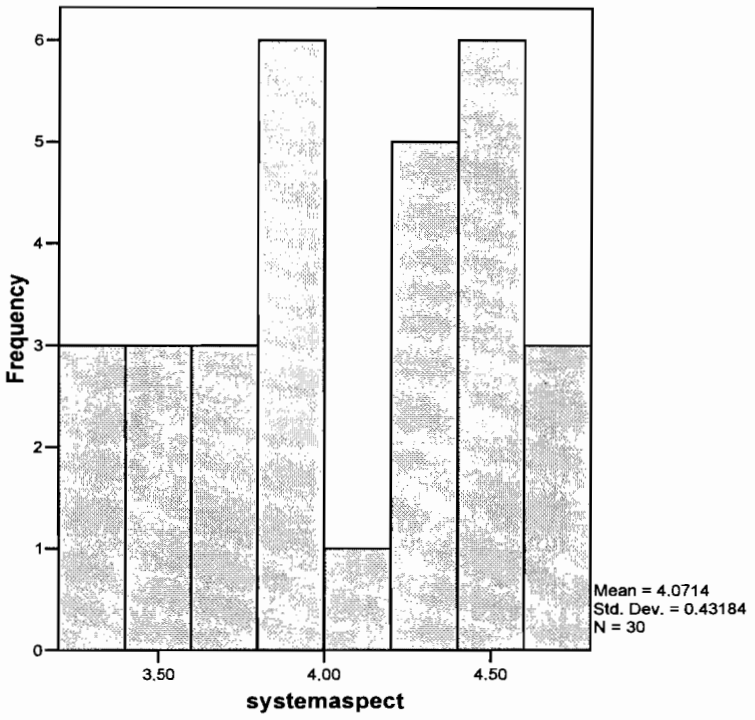

Chart 5.6: System Aspect

5.5 Summary

From the testing and evaluation conducted, the prototype fulfils the requirements needed the mobile user; refer to table 5.1 which show the reliability for this study. However, improvements has to be made for the prototype to be more user friendly by adding some images and colors to the prototype which needs a graphics designer. The prototype still needs to be published on Internet for further testing and improvements and real usage.

CHAPTER 6

CONCLUSION AND RECOMMENDED FURTHER STUDY

6.1 Introduction

This chapter will conclude, recommends and suggest additional features for this study. This includes contribution of study, problems and limitations encountered during the development of this project, then the recommendations and future work. Finally, this chapter will be ended with conclusion.

6.2 Contribution of Study

Lab assistants will be able to execute computer command remotely from a distant or another place over networked PCs inside the *UUM* labs at very low cost by using *PC Controller*. Any person can use the system to control his/her PC and the most important benefit of using *PC controller* is the effort saving because the assistants will done their operations without any effort in labs case which it is that they must go to each PC to execute an action her the effort will be lose but when we use the PC controller every thing done in the easiest way.

6.3 Problems and Limitations

- The main problem that I already faced was the time delay when the action sent from the mobile to the pc through the Bluetooth it was need a lot of time to execute the action.

- And the most problem it will has a conflict between the commands if we didn't remove the previous action it will conflict the new one

- And also when I was doing this research I faced problem to find literature review on the same topic because all the previous was on using IR and the networking controlling using the server.

- Regarding the limitation, one limitation that we faced which it is: How to make sure that the action that I execute it from a distance is already done for example the shutdown action without going to the PC? Distance for Bluetooth it is short 10 m.

- And also if there is software to show all the operation that will do on the network.

6.4 Recommendations and Future Work

After accomplishing the main phase of the project, there is a suggest additional features that if added will improve the quality and flexibility of this project, It is worth mentioning that the main objective in this study is to introduce new ideas for controlling the computers to be used in educational applications and other fields.

As an extension of this project and because the technology grows, may need to modify the code to fulfill these changes, so may suggest the followings as predict Table future aspects that may be added to the goals accomplished in this work.

- As an extension of this project and because as the sound technology advances, it is possible to control the computers by sound just by sending a voice message and the computer will compile and understand it then execute its command directly.

- It is possible to allow the computer to send comments about the command, and if there is an error it gives the user information about the type of the error and its source, and if it is true then send a reply that explain that the computer do its work successfully.

- That is seems probable to expand the project in order to cover the educational requirements, this option will help the lecturer to open a

specific files or send it to the students in easy way any time, and he can open his slides, view it, and move between it in easy way by his mobile from faraway distance, and use every application he may be need it in the class.

6.5 Conclusion

Controlling the network using Bluetooth technology provides a solution to control the network without wires, with low cost and high speed manner. Using Bluetooth technology makes the process unlimited because the divisions like walls can be passed and without need to have line of sight between the talking devices.

In *PC controller* application, the user can track all commands that sent to the computer through the rich text which shows all files that reached the folder. The user can send the commands and take the desired action from any device that is supported with Bluetooth technology, because of it is file transfer service, it is possible to use mobile phone or any computer to control the network. The user can guarantee the security, because the user must predefine the device which the user will send the files and give commands through it.

Visual c#.net is easy to use and provide the programmer with huge amount of libraries and built in functions that can use it. In this time it is possible to control the network in easy and interested way, from any computer or any mobile that supported by Bluetooth technology.

REFERENCES

Asahina .J, (2000), Bluetooth Protocol Stack, Troy XCD, Inc, from: <www.troyxcd.com>

Bluetooth, (Aug, 2006) Basics of Bluetooth wireless technology <http://www.bluetooth.com/Bluetooth/Technology/Basics.htm>

Broadcom Online, (March 2006). Basic Operations: Broadcom Wireless Adapter with Bluetooth® 2.0 + EDR Technology User Guide <http://update.broadcom.com/help/basic.htm>

Burns .G, (2006) Bluetooth Wireless Technology over UWB Demonstrated at Bluetooth SIG All Hands Meeting

Chau .P.Y.K. (1996) "An Empirical Assessment of a Modified Technology Acceptance Model", *Journal of Management Information Systems*, Vol.13 No.2, pp. 185-204.

Dasgupta .K, (2000), Bluetooth Protocol and Security Architecture Review, from : < http://www.cs.utk.edu/~dasgupta/bluetooth/>

Davis .F .D, (1989). Perceived usefulness, perceived ease of use, and user acceptance. *MIS Quarterly,* 13 (3), 319-340.

Davis .F.D, Bagozzi .R.P, and Warshaw, P.R. (1989): User Acceptance of Computer Technology: A Comparison of Two Theoretical Models. Management of Science; Aug 1989; 35, 8; ABI/INFORM Global pg.984

Dideles .M (2004). Bluetooth: A Technical Overview. Retrieved 7 October 2008 from <http://www.acm.org/crossroads/xrds9-4/blue.html.>

Dursch .A & Yen .D .C & Shih. D. (December 2003) *Bluetooth technology: an exploratory study of the analysis and implementation frameworks.* Miami University, Oxford and National Yunlin University of Science and Technology, Yunlin, Taiwan.

Haataja and Keijo M.J. 2006. Security in Bluetooth, WLAN and IrDA: a comparison. <http://www.cs.uku.fi/research/publications/reports/A-2006-1.pdf.>

Hair. et al. (2006), Mobilkom Austria, 'Geschichte der Mobilkom', Press Release Mobilkom Austria,. multivariate data analysis. Pearson prentice Hall Canada

Hattar. L, (2006), BLUETOOTH TECHNOLOGY/SECURITY, New York Institute of Technology (NYIT)

HP online, (2004), Hewlett-Packard Development Company, L.P.: Bluetooth wireless technology basics <h10032.www1.hp.com/ctg/Manual/c00186949.pdf>

IEEE online, (2004-Jan. 2005). What is Bluetooth? <http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1368913>

Informit Online, (Dec 2003). Introduction to Wireless Communication Systems: Advanced Techniques for Signal Reception. <http://www.informit.com/articles/article.aspx?p=102255>

Iogear, (2000), Discover the world of Bluetooth technology from: <www.iogear.com/guide/bluetoothguide.pdf>

Jakobsson .M and wetzel .S (2000), Security wireless in Bluetooth, USA

James .K, (2000), Bluetooth Architecture Overview <http://www.intel.com/technology/itj/q22000/pdf/art_1.pdf>

Jaquier .C & Drapel . K (February 11, 2005), Using Bluetoothr to Control a YaMoR Modular Robot, volum 1.0.

John .S, Robert .P, Edward .F, and Vijia .K, (2007), A Technology Review of Smart Sensors With Wireless Networks for Applications in Hazardous Work Environments

Kansal .A, (2002), A Bluetooth Primer

Karygiannis .T& Owens .L, (2002), Wireless Network Security: 802.11, Bluetooth and Handheld Devices

Kothari .R. (1985), Research Methodology: Methods and Techniques, Wiley Eastern, New Delhi

Law .C, Amar K. Mehta and Kai-yeung .S (2001) Performance of a new Bluetooth scatternet formation protocol

Lee, J .H & Bang H .J, (2006), Implementation and Design of PC Control System Using Mobile-Based Database

Magnuslator online, (August 22, 2007), from:< http://www.magnuslator.com/weblog/>

Mikhalenko .P .V, (2005). Developing Wireless Bluetooth Applications in J2ME. JDJ Volume 10, Issue <http://java.sys-con.com/read/47688.htm>

Misic .J & Misic .V .B & Chan .K .L. (February 2004) Performance of Bluetooth bridge scheduling algorithms, University of Manitoba, Canada & Hong King University of Science and Technology, China.

Nielsen .J. (June 26, 2006). Quantitative Studies: How Many Users to Test. Alertbox Retrieved 18 August, 2008, from http://www.useit.com/alertbox/quantitative_testing.html

Nokia Online, (April 2003) Bluetooth Technology Overview. <www.forum.nokia.com>.

Nordbotten .N .A & Aakvaag .N .D. (2004) Methods for service discovery in Bluetooth scatternets. Simula Research Laboratory, Lysaker & AAB Corporate Research, Billingstad, Norway.

Point .J.C, Ulbrich .M and Van .D .P, (2005), BROADBAND in Europe for All: A Multidisciplinary Approach, BREAD

Price .M, (2007) Fundamentals of Wireless Networking. New York: McGraw-hill/Irwin.

Rabaey .J et al., (2000) "PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking", IEEE Computer, Vol. 33, No. 7, pp. 42-48.

Rathi .S, (2000), Infrastructure: Bluetooth Protocol Architecture, Dedicated Systems Magazine

Rhodes .C, (2007), Bluetooth Security, from: <www.infosecwriters.com/text_resources/pdf/**Bluetooth**_CRhodes.pdf>

Rodrin .D, (2006), PERSONAL WIRELESS COMMUNICATIONS

Rumiana .K, Ani .B, Vesselin .G, and Ivilin .S.(2005). Application of Wireless Protocols Bluetooth and ZigBee in Telemetry System Development. <http://www.iit.bas.bg/PECR/55/30-38.pdf >

Sahbudin .R .K, Azhar .N, aris .I and Jeet .G .K, (2005), home and office environments using Bluetooth technology

Samppa .R, (1999), Wireless Communications in the Information Society

Sand, K. (1999) Telecommunications Software and Multimedia Laboratory Helsinki University of

Siegemund .F & Rohs .M. (2003) "Rendezvous Layer Protocols for Bluetooth-Enabled Smart Devices", *Personal and Ubiquitous Computing Journal.*

Silva .C (2006), Healthcare Computing & information management: IP DECT-the mobile technology of now. British journal, 23, 6

Softsea online. (2006). PC Remote Shutdown Pro. <http://www.softsea.com/review/PC-Remote-Shutdown_Pro.htm>

Song. A, (2001), Piconet II - Ad Hoc Multihop Wireless Networking Pico Communications OnLine, White Paper: Bluetooth and Bluetooth Internet Access Points. < www.picocommunications.com>.

Trap17 Online, (Aug 2006). Bluetooth Technology explained - An Overview of Bluetooth. <http://www.trap17.com/index.php/bluetooth-technology explained_t40377.html>

Vainio .J .T. 2000. Bluetooth Security. <http://www.niksula.hut.fi/~jiitv/bluesec.html.>

Vaishnavi .V and Kuechler .B (2004). Design Research in information system.. <http://www.isworld.org/Researchdesign/drisISworld.htm>

Verint,(2007), Nextiva Wireless Intelligent Edge Devices: An Introduction to Wireless Video

Whyte .W, (2005), Safe at Any Speed: Dedicated Short Range Communications (DSRC) and On-road Safety and Security

Zimmermann .L, (2004), Multimedia Applications in Education Conference (MApEC)

APPENDIX A

QUESTIONNAIRE

System to be evaluated:

PC controller application **(PCCA)**

Introduction:

This questionnaire consists of 19 questions in two parts

1. General information.

2. System aspects.

Please answer **ALL** questions in **ALL** parts

Part 1: General information

1. **Gender:**

[ ] male [ ] female

2. **Age:** _____ Years

3. **Educational background:**

[ ] Diploma [ ] Degree [ ] Master [ ] PhD

4. **How long you have been working as a lab assistant:**

[ ] 1 – 2 years

[ ] 2 – 3 years

[ ] More than 3 years

[] none of the above

5. **Which type of mobile you used _____ (e.g. Nokia N70)**

Part 2: System aspects

This part is intended to obtain your views on some aspects of the PC Controller Application (PCCA). Please mark [√] your answers.

1 = strongly disagree, 2 = Disagree, 3 = Natural, 4 = Agree, 5 = strongly agree

PERCEIVED USEFULNESS		1	2	3	4	5
6	Using **PCCA** will reduce effort	☐	☐	☐	☐	☐

7	Using **PCCA** will be reliability.	○	○	○	○	○
8	Using **PCCA** will reduce time	□	□	□	□	□
9	Using **PCCA** would increase my productivity	○	○	○	○	○
10	Using **PCCA** would enhance my effectiveness	□	□	□	□	□
11	Using **PCCA** would make it easier to do my tasks.	○	○	○	○	○
12	I would find **PCCA** useful to do my daily tasks	□	□	□	□	□
13	I would find **PCCA** secure.	○	○	○	○	○

PERCEIVED EASE OF USE		**1**	**2**	**3**	**4**	**5**
14	Learning to operate **PCCA** would be easy for me.	□	□	□	□	□
15	I would find it easy to get **PCCA** to do what I want it to do.	○	○	○	○	○
16	My interaction with **PCCA** would be clear and Understandable.	□	□	□	□	□
17	I would find **PCCA** to be flexible to interact with.	○	○	○	○	○
18	It will be easy for me to become skillful at using **PCCA.**	□	□	□	□	□
19	I would find **PCCA** easy to use.	○	○	○	○	○

Thank you for your cooperation and attention

APPENDIX B

COLLABORATION DIAGRAM

B.1 Search Bluetooth Collaboration Diagram

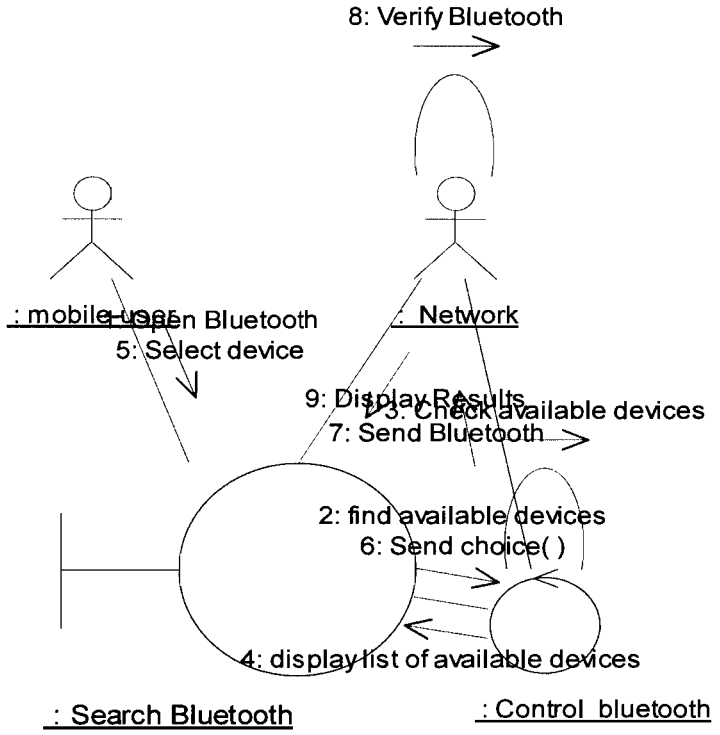

Figure B.1: Search Bluetooth Collaboration Diagram

B.2 Login Collaboration Diagram

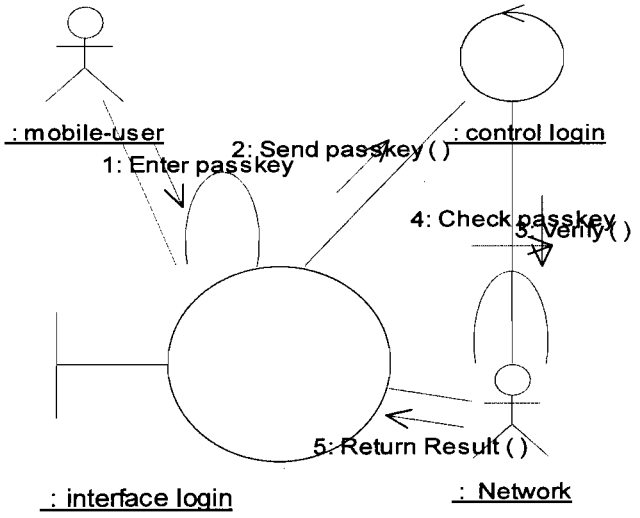

Figure B.2: Login Collaboration Diagram

B.3 Check Connection Collaboration Diagram

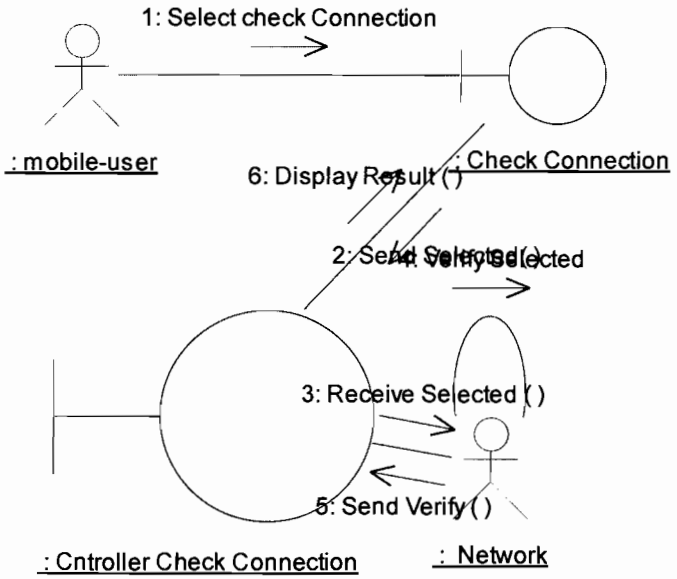

Figure B.3: Check Connection Collaboration Diagram

B.4 Shutdown Collaboration Diagram

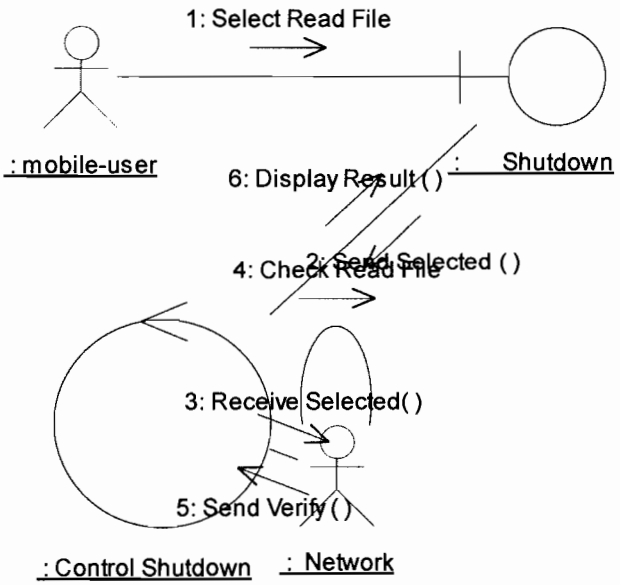

Figure B.4: Shutdown Collaboration Diagram

B.5 Logoff Collaboration Diagram

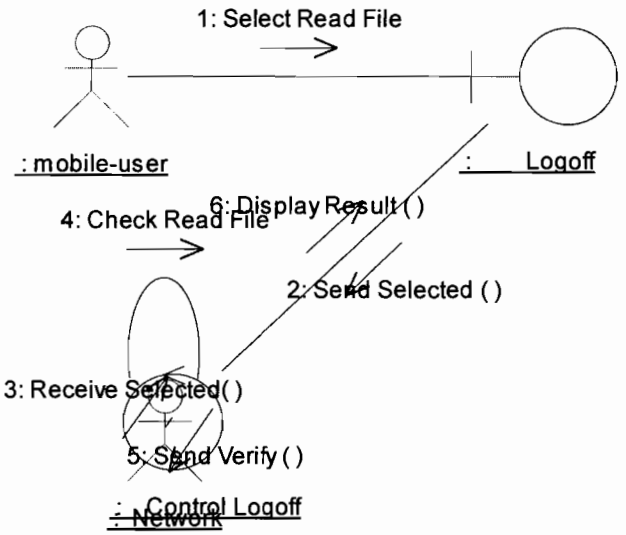

Figure B.5: Logoff Collaboration Diagram

B.6 Read file Collaboration Diagram

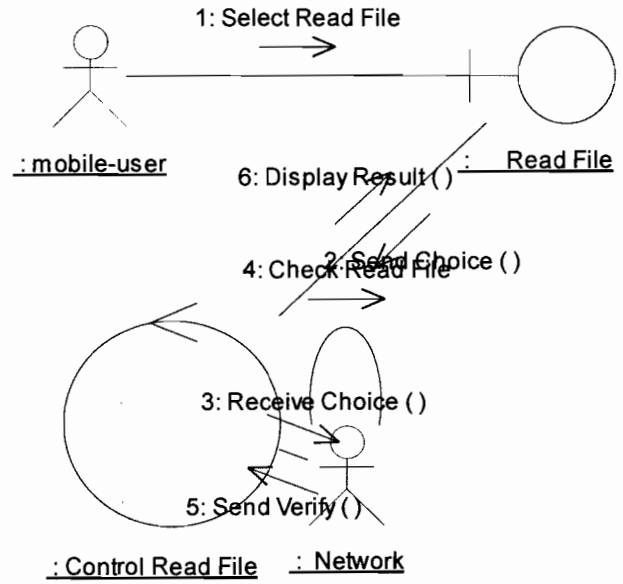

Figure B.6: Read File Collaboration Diagram

APPENDIX C

THE CODE

C.1 Server Code

```
using System;
using System.Drawing;
using System.Drawing.Printing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Resources;
using System.Management;
using System.Management.Instrumentation;
using System.Text;
using System.IO;
using Microsoft.Win32;
using mal;
namespace FileMonitoring
{
      /// <summary>
      /// Summary description for frmMonitor.
      /// </summary>
      public class frmMonitor : System.Windows.Forms.Form
      {



            private System.Windows.Forms.Label lblDirectoryPath;
            private System.Windows.Forms.TextBox txtDirectoryPath;
            private System.Windows.Forms.ListBox lsbEvents;
            private System.IO.FileSystemWatcher flsysWatcher;
            private System.Windows.Forms.ListBox listBox1;
            private System.Drawing.Printing.PrintDocument
printDocument1;
            private System.Windows.Forms.PrintDialog printDialog1;
        private Label label1;

            /// <summary>
            /// Required designer variable.
            /// </summary>
            private System.ComponentModel.Container components =
null;

            public frmMonitor()
            {
                  //
                  // Required for Windows Form Designer support
                  //
                  InitializeComponent();

                  //
                  // TODO: Add any constructor code after
InitializeComponent call
                  //
            }

            /// <summary>
            /// Clean up any resources being used.
```

```
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main()
		{
			Application.Run(new frmMonitor());

		}

		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
		System.ComponentModel.ComponentResourceManager resources
= new
System.ComponentModel.ComponentResourceManager(typeof(frmMonitor));
		this.lblDirectoryPath = new System.Windows.Forms.Label();
		this.txtDirectoryPath = new
System.Windows.Forms.TextBox();
		this.lsbEvents = new System.Windows.Forms.ListBox();
		this.flsysWatcher = new System.IO.FileSystemWatcher();
		this.listBox1 = new System.Windows.Forms.ListBox();
		this.printDocument1 = new
System.Drawing.Printing.PrintDocument();
		this.printDialog1 = new
System.Windows.Forms.PrintDialog();
		this.label1 = new System.Windows.Forms.Label();

((System.ComponentModel.ISupportInitialize)(this.flsysWatcher)).Begin
Init();
		this.SuspendLayout();
		//
		// lblDirectoryPath
		//
		this.lblDirectoryPath.BackColor =
System.Drawing.Color.SteelBlue;
		this.lblDirectoryPath.Location = new
System.Drawing.Point(0, 32);
		this.lblDirectoryPath.Name = "lblDirectoryPath";
		this.lblDirectoryPath.Size = new System.Drawing.Size(104,
16);
		this.lblDirectoryPath.TabIndex = 5;
		this.lblDirectoryPath.Text = "Monitored Directory ";
		this.lblDirectoryPath.Visible = false;
```

```
            //
            // txtDirectoryPath
            //
            this.txtDirectoryPath.BackColor =
System.Drawing.Color.SteelBlue;
            this.txtDirectoryPath.Location = new
System.Drawing.Point(104, 32);
            this.txtDirectoryPath.Name = "txtDirectoryPath";
            this.txtDirectoryPath.Size = new System.Drawing.Size(464,
20);
            this.txtDirectoryPath.TabIndex = 4;
            this.txtDirectoryPath.Visible = false;
            //
            // lsbEvents
            //
            this.lsbEvents.BackColor =
System.Drawing.Color.SteelBlue;
            this.lsbEvents.Font = new System.Drawing.Font("Times New
Roman", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.lsbEvents.ItemHeight = 19;
            this.lsbEvents.Location = new System.Drawing.Point(0,
352);
            this.lsbEvents.Name = "lsbEvents";
            this.lsbEvents.Size = new System.Drawing.Size(556, 118);
            this.lsbEvents.TabIndex = 7;
            this.lsbEvents.SelectedIndexChanged += new
System.EventHandler(this.lsbEvents_SelectedIndexChanged);
            //
            // flsysWatcher
            //
            this.flsysWatcher.EnableRaisingEvents = true;
            this.flsysWatcher.SynchronizingObject = this;
            this.flsysWatcher.Created += new
System.IO.FileSystemEventHandler(this.flsysWatcher_Created);
            this.flsysWatcher.Deleted += new
System.IO.FileSystemEventHandler(this.flsysWatcher_Deleted);
            this.flsysWatcher.Renamed += new
System.IO.RenamedEventHandler(this.flsysWatcher_Renamed);
            //
            // listBox1
            //
            this.listBox1.AccessibleRole =
System.Windows.Forms.AccessibleRole.MenuBar;
            this.listBox1.AllowDrop = true;
            this.listBox1.BackColor = System.Drawing.Color.SteelBlue;
            this.listBox1.Location = new System.Drawing.Point(0,
248);
            this.listBox1.Name = "listBox1";
            this.listBox1.Size = new System.Drawing.Size(556, 108);
            this.listBox1.TabIndex = 8;
            this.listBox1.Visible = false;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.BackColor =
System.Drawing.Color.CornflowerBlue;
            this.label1.Font = new System.Drawing.Font("Times New
Roman", 24F, System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point, ((byte)(178)));
```

```
            this.label1.ForeColor = System.Drawing.Color.Black;
            this.label1.Location = new System.Drawing.Point(61, -2);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(391, 36);
            this.label1.TabIndex = 9;
            this.label1.Text = "PC Controller For the Server ";
            this.label1.TextAlign =
System.Drawing.ContentAlignment.MiddleCenter;
            //
            // frmMonitor
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.BackColor = System.Drawing.Color.Gainsboro;
            this.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("$this.BackgroundImage"))
);
            this.ClientSize = new System.Drawing.Size(554, 467);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.listBox1);
            this.Controls.Add(this.lsbEvents);
            this.Controls.Add(this.lblDirectoryPath);
            this.Controls.Add(this.txtDirectoryPath);
            this.Name = "frmMonitor";
            this.Text = "Bluetooth";
            this.Load += new
System.EventHandler(this.frmMonitor_Load);

((System.ComponentModel.ISupportInitialize)(this.flsysWatcher)).EndIn
it();
            this.ResumeLayout(false);
            this.PerformLayout();

            }
            #endregion


            private void frmMonitor_Load(object sender,
System.EventArgs e)
            {
                  try
                  {
                DirectoryInfo dir1 = new DirectoryInfo(@"C:\Documents
and Settings\malek\My Documents\Bluetooth\inbox");
                        txtDirectoryPath.Text = dir1.ToString();
                        flsysWatcher.Path = txtDirectoryPath.Text;
                        flsysWatcher.EnableRaisingEvents = true;
                        txtDirectoryPath.Enabled = false;
                  }
                  catch(Exception ex)
                  {
                        ReportError(ex);
                  }
            }

/// <summary>
/// ////////////////////////////////////////////////////////

/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
```

```
            private void flsysWatcher_Created(object sender,
System.IO.FileSystemEventArgs e)
            {
                  try
                  {
                        string strEvent = "(" +  e.Name + ") has been
created in the directory";
                        lsbEvents.Items.Add(strEvent);

      /*if(System.IO.File.Exists("c:\\"+e.Name.ToString()))

      System.IO.File.Delete("c:\\"+e.Name.ToString());
                        else

      System.IO.File.Copy(txtDirectoryPath.Text + "\\" +
e.Name.ToString(),
                              "c:\\"+e.Name.ToString()); */

                ParseActions(e.Name);

                  }
                  catch(Exception ex)
                  {
                lsbEvents.Items.Add("'31a6");
                        ReportError(ex);
                  }

            }

            private void flsysWatcher_Deleted(object sender,
System.IO.FileSystemEventArgs e)
            {
                  try
                  {
                        string strEvent = "(" + e.Name + ") has been
deleted from the directory";
                        lsbEvents.Items.Add(strEvent);
                  }
                  catch(Exception ex)
                  {
                        ReportError(ex);
                  }

            }



            private void ParseActions(string strFileName)
            {
                  try
                  {
                        //I want to remove the extension, so let us
find the dot(.)
                        int nDotPosition =
strFileName.LastIndexOf('.');
                        //now the string is from the begenning up to
the dot position
                        string strActionName =
strFileName.Substring(0,strFileName.Length - (strFileName.Length -
nDotPosition) );
```

```
                        bool bHandled = true;
                        switch(strActionName.ToUpper())
                        {
                              //Case ---------- LogOff the computer
                                    case MobileActions.ActionLogOff:
                                lsbEvents.Items.Add("The Computer will
LogOff after 10 second ");
                                string path1 = txtDirectoryPath.Text +
"\\" + strFileName;
                                Win32Wrapper.Beep(2000, 1500);
                                mal.Class1 am = new Class1(2, path1);
                                    break;



//Case ---------- open empty notepad page.
                                    case MobileActions.ActionNotepad:
                                    lsbEvents.Items.Add("- Notepad
Action, Notepad will be opened now.");
                        Win32Wrapper.Beep(2000, 1500);
                                    Process.Start("Notepad.exe");
                        break;


//Case ---------- read document then open it.
                                    case MobileActions.ActionRead:
                                    {
                                          try{
                                string path2 = txtDirectoryPath.Text
+ "\\" + strFileName;
                                Win32Wrapper.Beep(1500, 940);
                                Form2 g = new Form2(path2);
                                                g.Show();
                                lsbEvents.Items.Add("The File
opened");
                                                }
                                          catch (Exception e)
                                          {

      Console.WriteLine("The file could not be read:");

      Console.WriteLine(e.Message);
                                          }
                                    }
                                    break;

//Case ---------- check if we now connect to internet or not.
                                    case MobileActions.ActionCheck:
                              Win32Wrapper.Beep(2000, 1500);
                              lsbEvents.Items.Add("The Connection
message showed");

      if(InternetCS.IsConnectedToInternet())
                                    MessageBox.Show("The Computer
Connected ", "Checking");
                                                      else
                                    MessageBox.Show("The Computer Not
Connected ", "Checking");

                                    break;
```

```
                    case MobileActions.ActionShutdown:
                        lsbEvents.Items.Add("The Computer will
Shutdown after 10 second ");
                        string path3 = txtDirectoryPath.Text + "\\" +
strFileName;
                        Win32Wrapper.Beep(2000, 1500);
                        mal.Class1 aml = new Class1(path3);

                                break;


                        }

                        //delete the file
                        if(bHandled)
                        {


      System.IO.File.Delete(txtDirectoryPath.Text + "\\" +
strFileName);
                        }
                    }
                    catch(Exception ex)
                    {
                lsbEvents.Items.Add("Error");
                        ReportError(ex);
                    }

            }



            private void flsysWatcher_Renamed(object sender,
System.IO.RenamedEventArgs e)
            {
                  try
                  {
                        string strEvent = "(" +  e.Name + ") has been
renamed in the directory";
                        lsbEvents.Items.Add(strEvent);
                        ParseActions(e.Name);
                  }
                  catch(Exception ex)
                  {
                        ReportError(ex);
                  }

            }

            private void ReportError(Exception ex)
            {
                  lsbEvents.Items.Add("Error:" +  ex.Message );
            }


            protected static string FormatError(int number )
            {
                  try
                  {
                        const int FORMAT_MESSAGE_FROM_SYSTEM =
0x1000;
```

```
                        StringBuilder buffer =new StringBuilder(255);

      Win32Wrapper.FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM,
IntPtr.Zero, number, 0, buffer, buffer.Capacity, 0);
                              return buffer.ToString();
                  }
                  catch (Exception)
                  {
                        return "Unspecified error [" +
number.ToString() + "]";
                  }
            }

        private void lsbEvents_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }

      }


      public class MobileActions
      {
            private MobileActions()//private constructor to prevent
creating instances, this is static data class
            {
            }
            public const string ActionLogOff = "MBL_LOGOFF";

            public const string ActionNotepad = "MBL_NOTEPAD";

            public const string ActionRead="MBL_READ";

            public const string ActionCheck="MBL_CHECK";
        public const string ActionShutdown = "MBL_SHUTDOWN";

      }
}



using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Threading;
using System.Windows.Forms;

using Org.Mentalis.Utilities;

namespace mal
{
   public class Classl
    {
            [DllImport("user32.dll")]
        private static extern
bool SetForegroundWindow(IntPtr hWnd);
        [DllImport("user32.dll")]
```

```
        private static extern
bool ShowWindowAsync(IntPtr hWnd, int nCmdShow);
        [DllImport("user32.dll")]
        private static extern
bool IsIconic(IntPtr hWnd);

        private const int SW_HIDE = 0;
        private const int SW_SHOWNORMAL = 1;
        private const int SW_SHOWMINIMIZED = 2;
        private const int SW_SHOWMAXIMIZED = 3;
        private const int SW_SHOWNOACTIVATE = 4;
        private const int SW_RESTORE = 9;
        private const int SW_SHOWDEFAULT = 10;

       public Class1(string s)
       {
           System.IO.File.Delete(s);
           System.Windows.Forms.Application.Exit();

Org.Mentalis.Utilities.WindowsController.ExitWindows(RestartOptions.P
owerOff, false);

       }
       public Class1(int i,string s)
       {
           System.IO.File.Delete(s);
           System.Windows.Forms.Application.Exit();

Org.Mentalis.Utilities.WindowsController.ExitWindows(RestartOptions.L
ogOff, false);

       }

    }
}



using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.IO;
using System.Resources;
namespace FileMonitoring
{
    /// <summary>
    /// Summary description for Form2.
    /// </summary>
    public class Form2 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.RichTextBox fileLoadArea;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;
        string path;

        public Form2(string s)
```

```
        {
            path = s;
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();


            //
            // TODO: Add any constructor code after
InitializeComponent call
            //
        }

        private void ReadFileInfo(String filename)
        {
            try
            {

                fileLoadArea.Text = File.ReadAllText(filename);

            }
            catch (Exception e)
            {

                Console.WriteLine("Exception" + e.StackTrace);

            }
        }



        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.fileLoadArea = new
System.Windows.Forms.RichTextBox();
            this.SuspendLayout();
            //
            // fileLoadArea
            //
```

```
            this.fileLoadArea.BackColor =
System.Drawing.SystemColors.Info;
            this.fileLoadArea.Dock =
System.Windows.Forms.DockStyle.Fill;
            this.fileLoadArea.Font = new System.Drawing.Font("Times
New Roman", 24F, System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point, ((byte)(178)));
            this.fileLoadArea.ForeColor = System.Drawing.Color.Blue;
            this.fileLoadArea.Location = new System.Drawing.Point(0,
0);
            this.fileLoadArea.Name = "fileLoadArea";
            this.fileLoadArea.Size = new System.Drawing.Size(504,
266);
            this.fileLoadArea.TabIndex = 0;
            this.fileLoadArea.Text = "";
            this.fileLoadArea.TextChanged += new
System.EventHandler(this.fileLoadArea_TextChanged);
            //
            // Form2
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(504, 266);
            this.Controls.Add(this.fileLoadArea);
            this.Name = "Form2";
            this.Text = "Open The File";
            this.WindowState =
System.Windows.Forms.FormWindowState.Maximized;
            this.Load += new System.EventHandler(this.Form2_Load);
            this.ResumeLayout(false);

        }
        #endregion



        private void Form2_Load(object sender, System.EventArgs e)
        {
            // = "C:\\"+"MBL_READ.txt";
            ReadFileInfo(path);
        }

        private void fileLoadArea_TextChanged(object sender,
System.EventArgs e)
        {

        }

        private void button1_Click(object sender, System.EventArgs e)
        {

        }

    }
}
```

*** WindowsController.cs**

```
using System;
using System.Text;
using System.Diagnostics;
using System.Runtime.InteropServices;

namespace Org.Mentalis.Utilities {
	/// <summary>
	/// Specifies the type of restart options that an application
can use.
	/// </summary>
	public enum RestartOptions {
		/// <summary>
		/// Shuts down all processes running in the security
context of the process that called the ExitWindowsEx function. Then
it logs the user off.
		/// </summary>
		LogOff = 0,
		/// <summary>
		/// Shuts down the system and turns off the power. The
system must support the power-off feature.
		/// </summary>
		PowerOff = 8,
		/// <summary>
		/// Shuts down the system and then restarts the system.
		/// </summary>
		Reboot = 2,
		/// <summary>
		/// Shuts down the system to a point at which it is safe
to turn off the power. All file buffers have been flushed to disk,
and all running processes have stopped. If the system supports the
power-off feature, the power is also turned off.
		/// </summary>
		ShutDown = 1,
		/// <summary>
		/// Suspends the system.
		/// </summary>
		Suspend = -1,
		/// <summary>
		/// Hibernates the system.
		/// </summary>
		Hibernate = -2,
	}
	/// <summary>
	/// An LUID is a 64-bit value guaranteed to be unique only on
the system on which it was generated. The uniqueness of a locally
unique identifier (LUID) is guaranteed only until the system is
restarted.
	/// </summary>
	[StructLayout(LayoutKind.Sequential, Pack=1)]
	internal struct LUID {
		/// <summary>
		/// The low order part of the 64 bit value.
		/// </summary>
		public int LowPart;
		/// <summary>
		/// The high order part of the 64 bit value.
		/// </summary>
		public int HighPart;
	}
```

```
        /// <summary>
        /// The LUID_AND_ATTRIBUTES structure represents a locally
unique identifier (LUID) and its attributes.
        /// </summary>
        [StructLayout(LayoutKind.Sequential, Pack=1)]
        internal struct LUID_AND_ATTRIBUTES {
                /// <summary>
                /// Specifies an LUID value.
                /// </summary>
                public LUID pLuid;
                /// <summary>
                /// Specifies attributes of the LUID. This value contains
up to 32 one-bit flags. Its meaning is dependent on the definition
and use of the LUID.
                /// </summary>
                public int Attributes;
        }
        /// <summary>
        /// The TOKEN_PRIVILEGES structure contains information about a
set of privileges for an access token.
        /// </summary>
        [StructLayout(LayoutKind.Sequential, Pack=1)]
        internal struct TOKEN_PRIVILEGES {
                /// <summary>
                /// Specifies the number of entries in the Privileges
array.
                /// </summary>
                public int PrivilegeCount ;
                /// <summary>
                /// Specifies an array of LUID_AND_ATTRIBUTES structures.
Each structure contains the LUID and attributes of a privilege.
                /// </summary>
                public LUID_AND_ATTRIBUTES Privileges ;
        }
        /// <summary>
        /// Implements methods to exit Windows.
        /// </summary>
        public class WindowsController {
                /// <summary>Required to enable or disable the privileges
in an access token.</summary>
                private const int TOKEN_ADJUST_PRIVILEGES = 0x20;
                /// <summary>Required to query an access token.</summary>
                private const int TOKEN_QUERY = 0x8;
                /// <summary>The privilege is enabled.</summary>
                private const int SE_PRIVILEGE_ENABLED = 0x2;
                /// <summary>Specifies that the function should search
the system message-table resource(s) for the requested
message.</summary>
                private const int FORMAT_MESSAGE_FROM_SYSTEM = 0x1000;
                /// <summary>Forces processes to terminate. When this
flag is set, the system does not send the WM_QUERYENDSESSION and
WM_ENDSESSION messages. This can cause the applications to lose data.
Therefore, you should only use this flag in an emergency.</summary>
                private const int EWX_FORCE = 4;
                /// <summary>
                /// The LoadLibrary function maps the specified
executable module into the address space of the calling process.
                /// </summary>
                /// <param name="lpLibFileName">Pointer to a null-
terminated string that names the executable module (either a .dll or
.exe file). The name specified is the file name of the module and is
```

```
not related to the name stored in the library module itself, as
specified by the LIBRARY keyword in the module-definition (.def)
file.</param>
            /// <returns>If the function succeeds, the return value
is a handle to the module.<br></br><br>If the function fails, the
return value is NULL. To get extended error information, call
Marshal.GetLastWin32Error.</br></returns>
            [ DllImport( "kernel32.dll", EntryPoint="LoadLibraryA",
CharSet=CharSet.Ansi )]
            private static extern IntPtr LoadLibrary(string
lpLibFileName);
            /// <summary>
            /// The FreeLibrary function decrements the reference
count of the loaded dynamic-link library (DLL). When the reference
count reaches zero, the module is unmapped from the address space of
the calling process and the handle is no longer valid.
            /// </summary>
            /// <param name="hLibModule">Handle to the loaded DLL
module. The LoadLibrary or GetModuleHandle function returns this
handle.</param>
            /// <returns>If the function succeeds, the return value
is nonzero.<br></br><br>If the function fails, the return value is
zero. To get extended error information, call
Marshal.GetLastWin32Error.</br></returns>
            [ DllImport( "kernel32.dll", EntryPoint="FreeLibrary",
CharSet=CharSet.Ansi )]
            private static extern int FreeLibrary(IntPtr hLibModule);
            /// <summary>
            /// The GetProcAddress function retrieves the address of
an exported function or variable from the specified dynamic-link
library (DLL).
            /// </summary>
            /// <param name="hModule">Handle to the DLL module that
contains the function or variable. The LoadLibrary or GetModuleHandle
function returns this handle.</param>
            /// <param name="lpProcName">Pointer to a null-terminated
string containing the function or variable name, or the function's
ordinal value. If this parameter is an ordinal value, it must be in
the low-order word; the high-order word must be zero.</param>
            /// <returns>If the function succeeds, the return value
is the address of the exported function or variable.<br></br><br>If
the function fails, the return value is NULL. To get extended error
information, call Marshal.GetLastWin32Error.</br></returns>
            [ DllImport( "kernel32.dll", EntryPoint="GetProcAddress",
CharSet=CharSet.Ansi )]
            private static extern IntPtr GetProcAddress(IntPtr
hModule, string lpProcName);
            /// <summary>
            /// The SetSuspendState function suspends the system by
shutting power down. Depending on the Hibernate parameter, the system
either enters a suspend (sleep) state or hibernation (S4). If the
ForceFlag parameter is TRUE, the system suspends operation
immediately; if it is FALSE, the system requests permission from all
applications and device drivers before doing so.
            /// </summary>
            /// <param name="Hibernate">Specifies the state of the
system. If TRUE, the system hibernates. If FALSE, the system is
suspended.</param>
            /// <param name="ForceCritical">Forced suspension. If
TRUE, the function broadcasts a PBT_APMSUSPEND event to each
application and driver, then immediately suspends operation. If
```

```
FALSE, the function broadcasts a PBT_APMQUERYSUSPEND event to each
application to request permission to suspend operation.</param>
            /// <param name="DisableWakeEvent">If TRUE, the system
disables all wake events. If FALSE, any system wake events remain
enabled.</param>
            /// <returns>If the function succeeds, the return value
is nonzero.<br></br><br>If the function fails, the return value is
zero. To get extended error information, call
Marshal.GetLastWin32Error.</br></returns>
            [ DllImport( "powrprof.dll",
EntryPoint="SetSuspendState", CharSet=CharSet.Ansi )]
            private static extern int SetSuspendState(int Hibernate,
int ForceCritical, int DisableWakeEvent);
            /// <summary>
            /// The OpenProcessToken function opens the access token
associated with a process.
            /// </summary>
            /// <param name="ProcessHandle">Handle to the process
whose access token is opened.</param>
            /// <param name="DesiredAccess">Specifies an access mask
that specifies the requested types of access to the access token.
These requested access types are compared with the token's
discretionary access-control list (DACL) to determine which accesses
are granted or denied.</param>
            /// <param name="TokenHandle">Pointer to a handle
identifying the newly-opened access token when the function
returns.</param>
            /// <returns>If the function succeeds, the return value
is nonzero.<br></br><br>If the function fails, the return value is
zero. To get extended error information, call
Marshal.GetLastWin32Error.</br></returns>
            [ DllImport( "advapi32.dll",
EntryPoint="OpenProcessToken", CharSet=CharSet.Ansi )]
            private static extern int OpenProcessToken(IntPtr
ProcessHandle, int DesiredAccess, ref IntPtr TokenHandle);
            /// <summary>
            /// The LookupPrivilegeValue function retrieves the
locally unique identifier (LUID) used on a specified system to
locally represent the specified privilege name.
            /// </summary>
            /// <param name="lpSystemName">Pointer to a null-
terminated string specifying the name of the system on which the
privilege name is looked up. If a null string is specified, the
function attempts to find the privilege name on the local
system.</param>
            /// <param name="lpName">Pointer to a null-terminated
string that specifies the name of the privilege, as defined in the
Winnt.h header file. For example, this parameter could specify the
constant SE_SECURITY_NAME, or its corresponding string,
"SeSecurityPrivilege".</param>
            /// <param name="lpLuid">Pointer to a variable that
receives the locally unique identifier by which the privilege is
known on the system, specified by the lpSystemName parameter.</param>
            /// <returns>If the function succeeds, the return value
is nonzero.<br></br><br>If the function fails, the return value is
zero. To get extended error information, call
Marshal.GetLastWin32Error.</br></returns>
            [ DllImport( "advapi32.dll",
EntryPoint="LookupPrivilegeValueA", CharSet=CharSet.Ansi )]
            private static extern int LookupPrivilegeValue(string
lpSystemName, string lpName, ref LUID lpLuid);
```

```
            /// <summary>
            /// The AdjustTokenPrivileges function enables or
disables privileges in the specified access token. Enabling or
disabling privileges in an access token requires
TOKEN_ADJUST_PRIVILEGES access.
            /// </summary>
            /// <param name="TokenHandle">Handle to the access token
that contains the privileges to be modified. The handle must have
TOKEN_ADJUST_PRIVILEGES access to the token. If the PreviousState
parameter is not NULL, the handle must also have TOKEN_QUERY
access.</param>
            /// <param name="DisableAllPrivileges">Specifies whether
the function disables all of the token's privileges. If this value is
TRUE, the function disables all privileges and ignores the NewState
parameter. If it is FALSE, the function modifies privileges based on
the information pointed to by the NewState parameter.</param>
            /// <param name="NewState">Pointer to a TOKEN_PRIVILEGES
structure that specifies an array of privileges and their attributes.
If the DisableAllPrivileges parameter is FALSE, AdjustTokenPrivileges
enables or disables these privileges for the token. If you set the
SE_PRIVILEGE_ENABLED attribute for a privilege, the function enables
that privilege; otherwise, it disables the privilege. If
DisableAllPrivileges is TRUE, the function ignores this
parameter.</param>
            /// <param name="BufferLength">Specifies the size, in
bytes, of the buffer pointed to by the PreviousState parameter. This
parameter can be zero if the PreviousState parameter is NULL.</param>
            /// <param name="PreviousState">Pointer to a buffer that
the function fills with a TOKEN_PRIVILEGES structure that contains
the previous state of any privileges that the function modifies. This
parameter can be NULL.</param>
            /// <param name="ReturnLength">Pointer to a variable that
receives the required size, in bytes, of the buffer pointed to by the
PreviousState parameter. This parameter can be NULL if PreviousState
is NULL.</param>
            /// <returns>If the function succeeds, the return value
is nonzero. To determine whether the function adjusted all of the
specified privileges, call Marshal.GetLastWin32Error.</returns>
            [ DllImport( "advapi32.dll",
EntryPoint="AdjustTokenPrivileges", CharSet=CharSet.Ansi )]
            private static extern int AdjustTokenPrivileges(IntPtr
TokenHandle, int DisableAllPrivileges, ref TOKEN_PRIVILEGES NewState,
int BufferLength, ref TOKEN_PRIVILEGES PreviousState, ref int
ReturnLength);
            /// <summary>
            /// The ExitWindowsEx function either logs off the
current user, shuts down the system, or shuts down and restarts the
system. It sends the WM_QUERYENDSESSION message to all applications
to determine if they can be terminated.
            /// </summary>
            /// <param name="uFlags">Specifies the type of
shutdown.</param>
            /// <param name="dwReserved">This parameter is
ignored.</param>
            /// <returns>If the function succeeds, the return value
is nonzero.<br></br><br>If the function fails, the return value is
zero. To get extended error information, call
Marshal.GetLastWin32Error.</br></returns>
            [ DllImport( "user32.dll", EntryPoint="ExitWindowsEx",
CharSet=CharSet.Ansi )]
```

```
            private static extern int ExitWindowsEx(int uFlags, int
dwReserved);
            /// <summary>
            /// The FormatMessage function formats a message string.
The function requires a message definition as input. The message
definition can come from a buffer passed into the function. It can
come from a message table resource in an already-loaded module. Or
the caller can ask the function to search the system's message table
resource(s) for the message definition. The function finds the
message definition in a message table resource based on a message
identifier and a language identifier. The function copies the
formatted message text to an output buffer, processing any embedded
insert sequences if requested.
            /// </summary>
            /// <param name="dwFlags">Specifies aspects of the
formatting process and how to interpret the lpSource parameter. The
low-order byte of dwFlags specifies how the function handles line
breaks in the output buffer. The low-order byte can also specify the
maximum width of a formatted output line.</param>
            /// <param name="lpSource">Specifies the location of the
message definition. The type of this parameter depends upon the
settings in the dwFlags parameter.</param>
            /// <param name="dwMessageId">Specifies the message
identifier for the requested message. This parameter is ignored if
dwFlags includes FORMAT_MESSAGE_FROM_STRING.</param>
            /// <param name="dwLanguageId">Specifies the language
identifier for the requested message. This parameter is ignored if
dwFlags includes FORMAT_MESSAGE_FROM_STRING.</param>
            /// <param name="lpBuffer">Pointer to a buffer for the
formatted (and null-terminated) message. If dwFlags includes
FORMAT_MESSAGE_ALLOCATE_BUFFER, the function allocates a buffer using
the LocalAlloc function, and places the pointer to the buffer at the
address specified in lpBuffer.</param>
            /// <param name="nSize">If the
FORMAT_MESSAGE_ALLOCATE_BUFFER flag is not set, this parameter
specifies the maximum number of TCHARs that can be stored in the
output buffer. If FORMAT_MESSAGE_ALLOCATE_BUFFER is set, this
parameter specifies the minimum number of TCHARs to allocate for an
output buffer. For ANSI text, this is the number of bytes; for
Unicode text, this is the number of characters.</param>
            /// <param name="Arguments">Pointer to an array of values
that are used as insert values in the formatted message. A %1 in the
format string indicates the first value in the Arguments array; a %2
indicates the second argument; and so on.</param>
            /// <returns>If the function succeeds, the return value
is the number of TCHARs stored in the output buffer, excluding the
terminating null character.<br></br><br>If the function fails, the
return value is zero. To get extended error information, call
Marshal.GetLastWin32Error.</br></returns>
            [ DllImport( "user32.dll", EntryPoint="FormatMessageA",
CharSet=CharSet.Ansi )]
            private static extern int FormatMessage(int dwFlags,
IntPtr lpSource, int dwMessageId, int dwLanguageId, StringBuilder
lpBuffer, int nSize, int Arguments);
            /// <summary>
            /// Exits windows (and tries to enable any required
access rights, if necesarry).
            /// </summary>
            /// <param name="how">One of the RestartOptions values
that specifies how to exit windows.</param>
```

```
            /// <param name="force">True if the exit has to be
forced, false otherwise.</param>
            /// <exception cref="PrivilegeException">There was an
error while requesting a required privilege.</exception>
            /// <exception cref="PlatformNotSupportedException">The
requested exit method is not supported on this platform.</exception>
            public static void ExitWindows(RestartOptions how , bool
force ) {
                  switch(how) {
                        case RestartOptions.Suspend:
                              SuspendSystem(false, force);
                              break;
                        case RestartOptions.Hibernate:
                              SuspendSystem(true, force);
                              break;
                        default:
                              ExitWindows((int)how, force);
                              break;
                  }
            }
            /// <summary>
            /// Exits windows (and tries to enable any required
access rights, if necesarry).
            /// </summary>
            /// <param name="how">One of the RestartOptions values
that specifies how to exit windows.</param>
            /// <param name="force">True if the exit has to be
forced, false otherwise.</param>
            /// <remarks>This method cannot hibernate or suspend the
system.</remarks>
            /// <exception cref="PrivilegeException">There was an
error while requesting a required privilege.</exception>
            protected static void ExitWindows(int how , bool force )
{
                  EnableToken("SeShutdownPrivilege");
                  if (force )
                        how = how | EWX_FORCE;
                  if (ExitWindowsEx(how, 0) == 0)
                        throw new
PrivilegeException(FormatError(Marshal.GetLastWin32Error()));
            }
            /// <summary>
            /// Tries to enable the specified privilege.
            /// </summary>
            /// <param name="privilege">The privilege to
enable.</param>
            /// <exception cref="PrivilegeException">There was an
error while requesting a required privilege.</exception>
            protected static void EnableToken(string privilege ) {
                  if (!CheckEntryPoint("advapi32.dll",
"AdjustTokenPrivileges"))
                        return;
                  IntPtr tokenHandle = IntPtr.Zero;
                  LUID privilegeLUID = new LUID();
                  TOKEN_PRIVILEGES newPrivileges = new
TOKEN_PRIVILEGES();
                  TOKEN_PRIVILEGES tokenPrivileges ;
                  if
(OpenProcessToken(Process.GetCurrentProcess().Handle,
TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY, ref tokenHandle) == 0)
```

```
                        throw new
PrivilegeException(FormatError(Marshal.GetLastWin32Error()));
                    if (LookupPrivilegeValue("", privilege, ref
privilegeLUID) == 0)
                        throw new
PrivilegeException(FormatError(Marshal.GetLastWin32Error()));
                    tokenPrivileges.PrivilegeCount = 1;
                    tokenPrivileges.Privileges.Attributes =
SE_PRIVILEGE_ENABLED;
                    tokenPrivileges.Privileges.pLuid = privilegeLUID;
                    int size = 4;
                    if (AdjustTokenPrivileges(tokenHandle, 0, ref
tokenPrivileges, 4 + (12 * tokenPrivileges.PrivilegeCount), ref
newPrivileges, ref size) == 0)
                        throw new
PrivilegeException(FormatError(Marshal.GetLastWin32Error()));
            }
            /// <summary>
            /// Suspends or hibernates the system.
            /// </summary>
            /// <param name="hibernate">True if the system has to
hibernate, false if the system has to be suspended.</param>
            /// <param name="force">True if the exit has to be
forced, false otherwise.</param>
            /// <exception cref="PlatformNotSupportedException">The
requested exit method is not supported on this platform.</exception>
            protected static void SuspendSystem(bool hibernate , bool
force  ){
                    if (!CheckEntryPoint("powrprof.dll",
"SetSuspendState"))
                        throw new PlatformNotSupportedException("The
SetSuspendState method is not supported on this system!");
                    SetSuspendState((int)(hibernate ? 1 : 0),
(int)(force ? 1 : 0), 0);
            }
            /// <summary>
            /// Checks whether a specified method exists on the local
computer.
            /// </summary>
            /// <param name="library">The library that holds the
method.</param>
            /// <param name="method">The entry point of the requested
method.</param>
            /// <returns>True if the specified method is present,
false otherwise.</returns>
            protected static bool CheckEntryPoint(string library ,
string method ) {
                    IntPtr  libPtr = LoadLibrary(library);
                    if (!libPtr.Equals(IntPtr.Zero)) {
                        if (!GetProcAddress(libPtr,
method).Equals(IntPtr.Zero)) {
                            FreeLibrary(libPtr);
                            return true;
                        }
                        FreeLibrary(libPtr);
                    }
                    return false;
            }
            /// <summary>
            /// Formats an error number into an error message.
            /// </summary>
```

```
            /// <param name="number">The error number to
convert.</param>
            /// <returns>A string representation of the specified
error number.</returns>
            protected static string FormatError(int number ) {
                  StringBuilder buffer =new StringBuilder(255);
                  FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM,
IntPtr.Zero, number, 0, buffer, buffer.Capacity, 0);
                  return buffer.ToString();
            }
      }
      /// <summary>
      /// The exception that is thrown when an error occures when
requesting a specific privilege.
      /// </summary>
      public class PrivilegeException : Exception {
            /// <summary>
            /// Initializes a new instance of the PrivilegeException
class.
            /// </summary>
            public PrivilegeException () : base() {}
            /// <summary>
            /// Initializes a new instance of the PrivilegeException
class with a specified error message.
            /// </summary>
            /// <param name="message">The message that describes the
error.</param>
            public PrivilegeException (string message )
:base(message) {}
      }
}
```

C.2 Code for Client

```
using System;
using System.Drawing;
using System.Drawing.Printing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Resources;
using System.Management;
using System.Management.Instrumentation;
using System.Text;
using System.IO;
using Microsoft.Win32;
using mal;
namespace FileMonitoring
{
      /// <summary>
      /// Summary description for frmMonitor.
      /// </summary>
      public class frmMonitor : System.Windows.Forms.Form
      {
```

```
		private System.Windows.Forms.Label lblDirectoryPath;
		private System.Windows.Forms.TextBox txtDirectoryPath;
		private System.Windows.Forms.ListBox lsbEvents;
		private System.IO.FileSystemWatcher flsysWatcher;
		private System.Windows.Forms.ListBox listBox1;
		private System.Drawing.Printing.PrintDocument
printDocument1;
		private System.Windows.Forms.PrintDialog printDialog1;
	private Label label1;

		/// <summary>
		/// Required designer variable.
		/// </summary>
		private System.ComponentModel.Container components =
null;

		public frmMonitor()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

			//
			// TODO: Add any constructor code after
InitializeComponent call
			//
		}

		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if(components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main()
		{
			Application.Run(new frmMonitor());

		}

		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
```

```
            {
            System.ComponentModel.ComponentResourceManager resources
= new
System.ComponentModel.ComponentResourceManager(typeof(frmMonitor));
            this.lblDirectoryPath = new System.Windows.Forms.Label();
            this.txtDirectoryPath = new
System.Windows.Forms.TextBox();
            this.lsbEvents = new System.Windows.Forms.ListBox();
            this.flsysWatcher = new System.IO.FileSystemWatcher();
            this.listBox1 = new System.Windows.Forms.ListBox();
            this.printDocument1 = new
System.Drawing.Printing.PrintDocument();
            this.printDialog1 = new
System.Windows.Forms.PrintDialog();
            this.label1 = new System.Windows.Forms.Label();

((System.ComponentModel.ISupportInitialize)(this.flsysWatcher)).Begin
Init();
            this.SuspendLayout();
            //
            // lblDirectoryPath
            //
            this.lblDirectoryPath.BackColor =
System.Drawing.Color.SteelBlue;
            this.lblDirectoryPath.Location = new
System.Drawing.Point(0, 32);
            this.lblDirectoryPath.Name = "lblDirectoryPath";
            this.lblDirectoryPath.Size = new System.Drawing.Size(104,
16);
            this.lblDirectoryPath.TabIndex = 5;
            this.lblDirectoryPath.Text = "Monitored Directory ";
            this.lblDirectoryPath.Visible = false;
            //
            // txtDirectoryPath
            //
            this.txtDirectoryPath.BackColor =
System.Drawing.Color.SteelBlue;
            this.txtDirectoryPath.Location = new
System.Drawing.Point(104, 32);
            this.txtDirectoryPath.Name = "txtDirectoryPath";
            this.txtDirectoryPath.Size = new System.Drawing.Size(480,
20);
            this.txtDirectoryPath.TabIndex = 4;
            this.txtDirectoryPath.Visible = false;
            //
            // lsbEvents
            //
            this.lsbEvents.BackColor =
System.Drawing.Color.SteelBlue;
            this.lsbEvents.Font = new System.Drawing.Font("Times New
Roman", 12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.lsbEvents.ItemHeight = 19;
            this.lsbEvents.Location = new System.Drawing.Point(0,
352);
            this.lsbEvents.Name = "lsbEvents";
            this.lsbEvents.Size = new System.Drawing.Size(568, 118);
            this.lsbEvents.TabIndex = 7;
            this.lsbEvents.SelectedIndexChanged += new
System.EventHandler(this.lsbEvents_SelectedIndexChanged);
            //
```

```
            // flsysWatcher
            //
            this.flsysWatcher.EnableRaisingEvents = true;
            this.flsysWatcher.SynchronizingObject = this;
            this.flsysWatcher.Created += new
System.IO.FileSystemEventHandler(this.flsysWatcher_Created);
            this.flsysWatcher.Deleted += new
System.IO.FileSystemEventHandler(this.flsysWatcher_Deleted);
            this.flsysWatcher.Renamed += new
System.IO.RenamedEventHandler(this.flsysWatcher_Renamed);
            //
            // listBox1
            //
            this.listBox1.AccessibleRole =
System.Windows.Forms.AccessibleRole.MenuBar;
            this.listBox1.AllowDrop = true;
            this.listBox1.BackColor = System.Drawing.Color.SteelBlue;
            this.listBox1.Location = new System.Drawing.Point(0,
248);
            this.listBox1.Name = "listBox1";
            this.listBox1.Size = new System.Drawing.Size(568, 108);
            this.listBox1.TabIndex = 8;
            this.listBox1.Visible = false;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.BackColor =
System.Drawing.Color.CornflowerBlue;
            this.label1.Font = new System.Drawing.Font("Times New
Roman", 24F, System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point, ((byte)(178)));
            this.label1.ForeColor = System.Drawing.Color.Black;
            this.label1.Location = new System.Drawing.Point(61, -1);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(377, 36);
            this.label1.TabIndex = 10;
            this.label1.Text = "PC Controller For the Client";
            this.label1.TextAlign =
System.Drawing.ContentAlignment.MiddleCenter;
            //
            // frmMonitor
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.BackColor = System.Drawing.Color.Gainsboro;
            this.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("$this.BackgroundImage"))
);
            this.ClientSize = new System.Drawing.Size(568, 470);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.listBox1);
            this.Controls.Add(this.lsbEvents);
            this.Controls.Add(this.lblDirectoryPath);
            this.Controls.Add(this.txtDirectoryPath);
            this.Name = "frmMonitor";
            this.Text = "Bluetooth";
            this.Load += new
System.EventHandler(this.frmMonitor_Load);

((System.ComponentModel.ISupportInitialize)(this.flsysWatcher)).EndIn
it();
```

```
            this.ResumeLayout(false);
            this.PerformLayout();

            }
            #endregion


            private void frmMonitor_Load(object sender,
System.EventArgs e)
            {
                  try
                  {
                DirectoryInfo dir1 = new
DirectoryInfo(@"\\UUM\inbox");
                        txtDirectoryPath.Text = dir1.ToString();
                        flsysWatcher.Path = txtDirectoryPath.Text;
                        flsysWatcher.EnableRaisingEvents = true;
                        txtDirectoryPath.Enabled = false;
                  }
                  catch(Exception ex)
                  {
                        ReportError(ex);
                  }
            }

/// <summary>
/// ////////////////////////////////////////////////////////

/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>


            private void flsysWatcher_Created(object sender,
System.IO.FileSystemEventArgs e)
            {
                  try
                  {

                              ParseActions(e.Name);

                  }
                  catch(Exception ex)
                  {
                        ReportError(ex);
                  }

            }

            private void flsysWatcher_Deleted(object sender,
System.IO.FileSystemEventArgs e)
            {
                  try
                  {
                        string strEvent = "(" + e.Name + ") has been
deleted from the directory";
                        lsbEvents.Items.Add(strEvent);
                  }
                  catch(Exception ex)
                  {
                        ReportError(ex);
```

```
                }

            }


            private void ParseActions(string strFileName)
            {
                try
                {
                    //I want to remove the extension, so let us
find the dot(.)
                    int nDotPosition =
strFileName.LastIndexOf('.');
                    //now the string is from the begenning up to
the dot position
                    string strActionName =
strFileName.Substring(0,strFileName.Length - (strFileName.Length -
nDotPosition) );
                    switch(strActionName.ToUpper())
                    {
                        //Case ---------- LogOff the computer
                            case MobileActions.ActionLogOff:
                        string path1 = txtDirectoryPath.Text +
"\\" + strFileName;
                        Win32Wrapper.Beep(1500, 400);
                        mal.Class1 am = new Class1(2, path1);
                            break;


//Case ---------- open empty notepad page.
                            case MobileActions.ActionNotepad:
                            lsbEvents.Items.Add("- Notepad
Action, Notepad will be opened now.");
                    Win32Wrapper.Beep(1500,400);
                            Process.Start("Notepad.exe");
                    break;


//Case ---------- read document then open it.
                            case MobileActions.ActionRead:
                            {
                                try{
string path2 = txtDirectoryPath.Text + "\\" + strFileName;
                                Win32Wrapper.Beep(1500, 400);
                                Form2 g = new Form2(path2);
                                        g.Show();
                                lsbEvents.Items.Add("The File
opened");
                                    }
                                catch (Exception e)
                                {

      Console.WriteLine("The file could not be read:");

      Console.WriteLine(e.Message);
                                }
                            }
                            break;
```

```
                              //Case ---------- check if we now
connect to internet or not.
                                    case MobileActions.ActionCheck:
                              Win32Wrapper.Beep(1500, 400);
                              lsbEvents.Items.Add("The Connection
Message Showed");

      if(InternetCS.IsConnectedToInternet())

      MessageBox.Show("The Computer Connected with Server
","Checking");
                                                      else
                                    MessageBox.Show("The Computer Not
Connected with Server ", "Checking");


                                    break;

           case MobileActions.ActionShutdown:
          string path3 = txtDirectoryPath.Text + "\\" + strFileName;
                        Win32Wrapper.Beep(1500, 400);
                        mal.Class1 aml = new Class1(path3);

                                    break;


                        }


                  }
                  catch(Exception ex)
                  {
                        ReportError(ex);
                  }

            }


            private void flsysWatcher_Renamed(object sender,
System.IO.RenamedEventArgs e)
            {
                  try
                  {
                        string strEvent = "(" +  e.Name + ") has been
renamed in the directory";
                        lsbEvents.Items.Add(strEvent);
                        ParseActions(e.Name);
                  }
                  catch(Exception ex)
                  {
                        ReportError(ex);
                  }

            }

            private void ReportError(Exception ex)
            {
                  lsbEvents.Items.Add("Error:" +  ex.Message );
            }
```

```
            protected static string FormatError(int number )
            {
                  try
                  {
                        const int FORMAT_MESSAGE_FROM_SYSTEM =
0x1000;
                        StringBuilder buffer =new StringBuilder(255);

      Win32Wrapper.FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM,
IntPtr.Zero, number, 0, buffer, buffer.Capacity, 0);
                              return buffer.ToString();
                  }
                  catch (Exception)
                  {
                        return "Unspecified error [" +
number.ToString() + "]";
                  }
            }

        private void lsbEvents_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }

      }


      public class MobileActions
      {
            private MobileActions()//private constructor to prevent
creating instances, this is static data class
            {
            }
            public const string ActionLogOff = "MBL_LOGOFF";

            public const string ActionNotepad = "MBL_NOTEPAD";

            public const string ActionRead="MBL_READ";

            public const string ActionCheck="MBL_CHECK";
          public const string ActionShutdown = "MBL_SHUTDOWN";

      }
}


using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.IO;
using System.Resources;
namespace FileMonitoring
{
    /// <summary>
    /// Summary description for Form2.
    /// </summary>
    public class Form2 : System.Windows.Forms.Form
```

```
    {
        private System.Windows.Forms.RichTextBox fileLoadArea;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;
        string path;

        public Form2(string s)
        {
            path = s;
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();


            //
            // TODO: Add any constructor code after
InitializeComponent call
            //
        }

        private void ReadFileInfo(String filename)
        {
            try
            {
                fileLoadArea.Text = File.ReadAllText(filename);

            }
            catch (Exception e)
            {
                Console.WriteLine("Exception" + e.StackTrace);
            }
        }


        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.fileLoadArea = new
System.Windows.Forms.RichTextBox();
```

```
            this.SuspendLayout();
            //
            // fileLoadArea
            //
            this.fileLoadArea.BackColor =
System.Drawing.SystemColors.Info;
            this.fileLoadArea.Dock =
System.Windows.Forms.DockStyle.Fill;
            this.fileLoadArea.Font = new System.Drawing.Font("Times
New Roman", 24F, System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point, ((byte)(178)));
            this.fileLoadArea.ForeColor = System.Drawing.Color.Blue;
            this.fileLoadArea.Location = new System.Drawing.Point(0,
0);
            this.fileLoadArea.Name = "fileLoadArea";
            this.fileLoadArea.Size = new System.Drawing.Size(504,
266);
            this.fileLoadArea.TabIndex = 0;
            this.fileLoadArea.Text = "";
            this.fileLoadArea.TextChanged += new
System.EventHandler(this.fileLoadArea_TextChanged);
            //
            // Form2
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(504, 266);
            this.Controls.Add(this.fileLoadArea);
            this.Name = "Form2";
            this.Text = "Open The File";
            this.WindowState =
System.Windows.Forms.FormWindowState.Maximized;
            this.Load += new System.EventHandler(this.Form2_Load);
            this.ResumeLayout(false);

        }
        #endregion


        private void Form2_Load(object sender, System.EventArgs e)
        {
            // = "C:\\"+"MBL_READ.txt";
            ReadFileInfo(path);
        }

        private void fileLoadArea_TextChanged(object sender,
System.EventArgs e)
        {

        }

        private void button1_Click(object sender, System.EventArgs e)
        {

        }

    }
}
```

APPENDIX D

CLASS DIAGRAM

D.1 Class Diagram

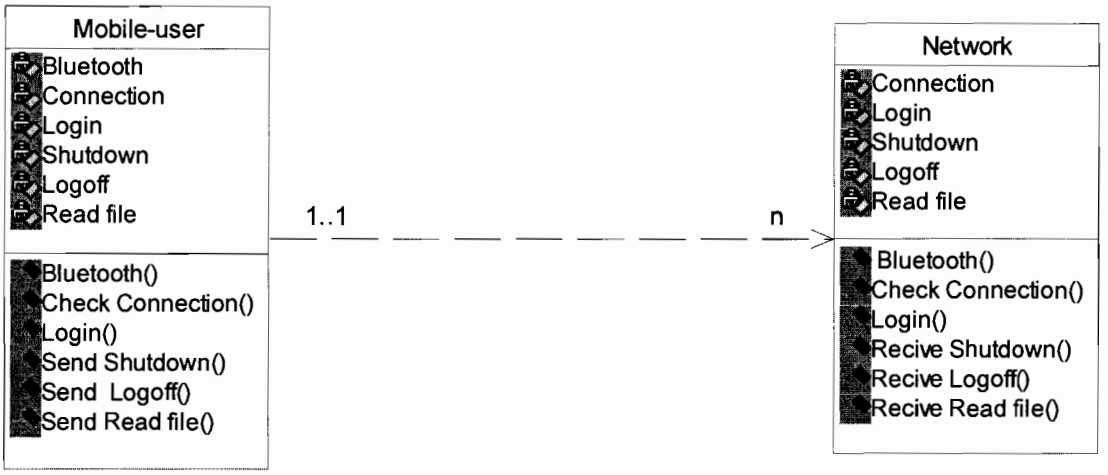

Figure D.1: mobile user and network class diagram

Figure D.1 shows the relationship between the main two objects in this study, the mobile user and the network where there is a link between them with elaborating their attributes and operations.

APPENDIX E

PROGRAM USED AND PREVIOUS EXPERIMENT

E.1 Bluetooth Software in the Server (BlueSoleil)

Figure E.1 shows the BlueSoleil Program this is Bluetooth software define the Bluetooth device in the server to let the server receive the files and do the Bluetooth exchange folder n the specific path in the server.

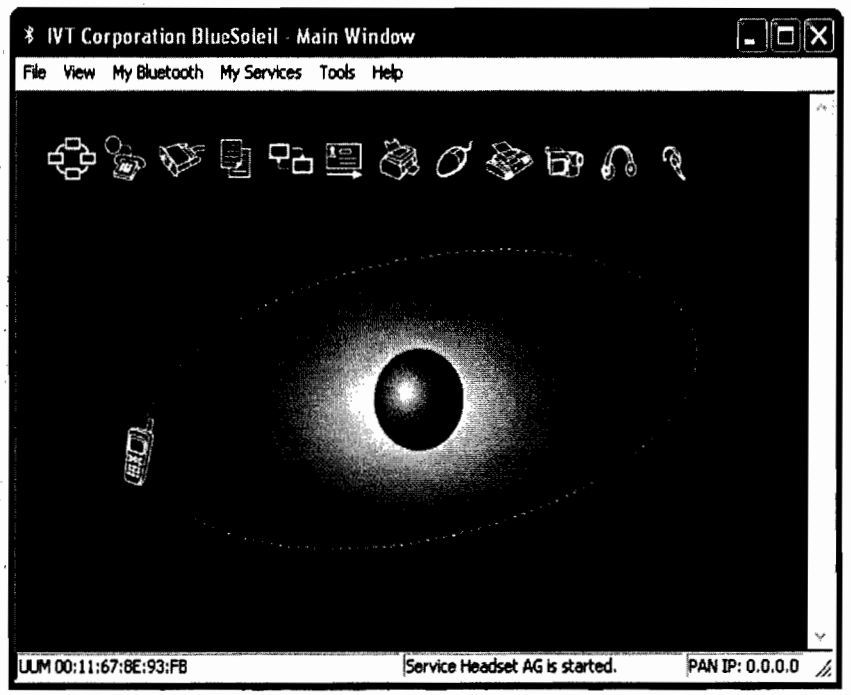

Figure E.1: BlueSoleil Program

BlueSoleil is Windows-based software from IVT that allows your Bluetooth enabled desktop or notebook computer to wirelessly connect to other Bluetooth enabled devices. BlueSoleil allows MS Windows users to wirelessly access a wide variety of Bluetooth enabled digital devices, such as cameras, mobile phones, and headsets, printers, and GPS receivers. You can also form networks and exchange data with other Bluetooth enabled computers or PDAs.

Platforms supported by BlueSoleil include:

- Windows 2000/XP
- Windows Vista

E.2 Open empty notepad (Notepad Action)

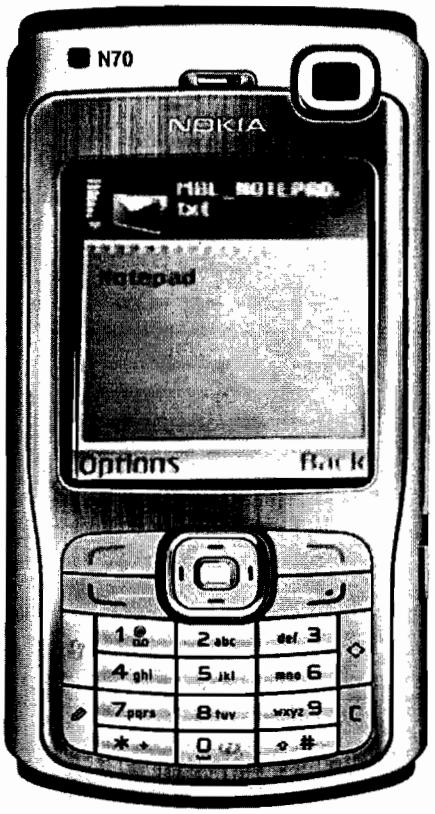

Figure E.2: Notepad File

This action just open empty new notepad page and view it at terminal, this action will be done if we send file its name ***MBL_NOTEPAD.TXT*** then directly will be opened in the server t will appear as it shown in Figure E.3.

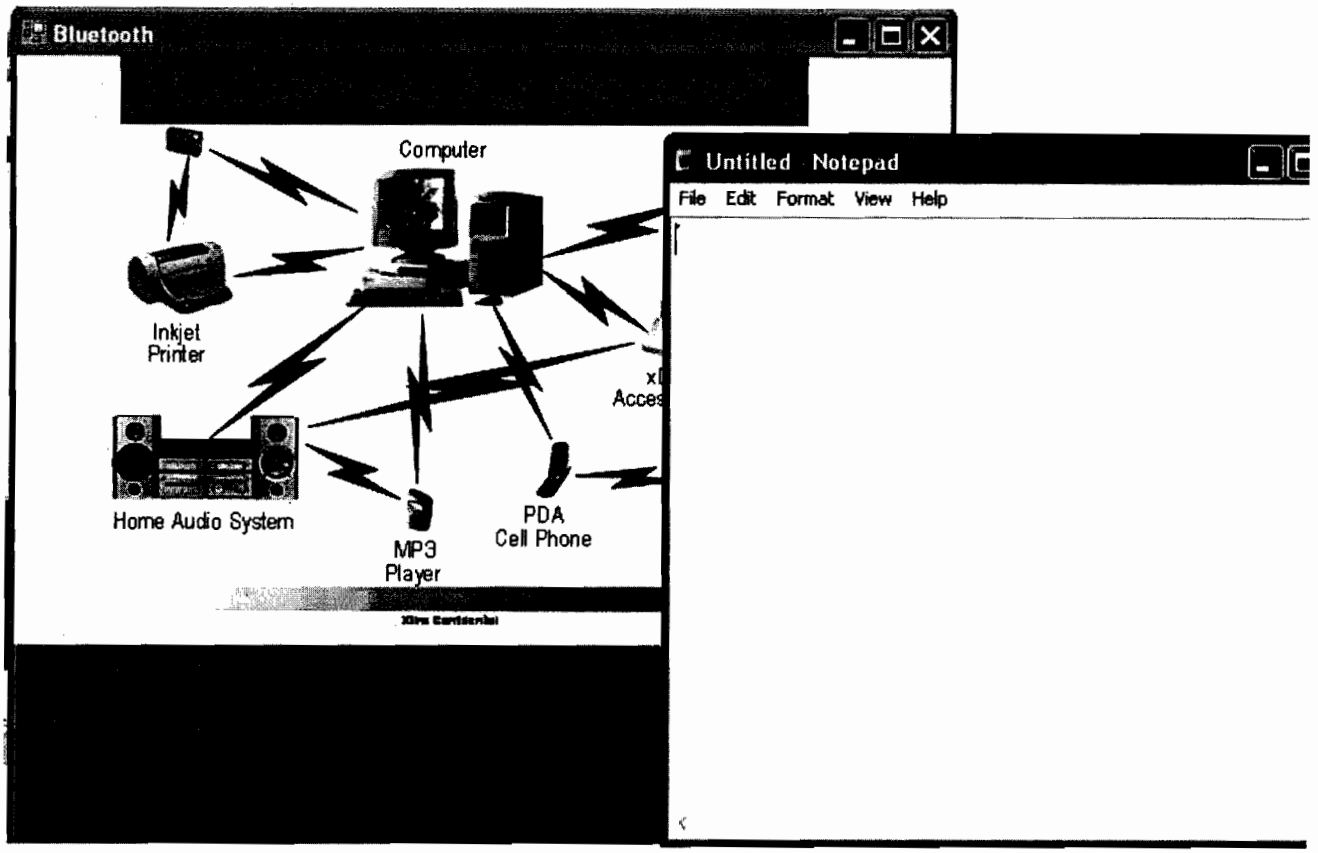

Figure E.3: Action in the server

As it shown in the Figure E.3, at the first the server should create a file in the specific path and shows the message (MBL_NOTEPAD.TXT) has been created in the directory, and then the server will shows message to tell the user Notepad will opened now, at the last when the action done the server will delete the file from the directory to ensure there are not conflict will happen if the user send the message again. At the same time the action will appear at the client site as it shown in the Figur88

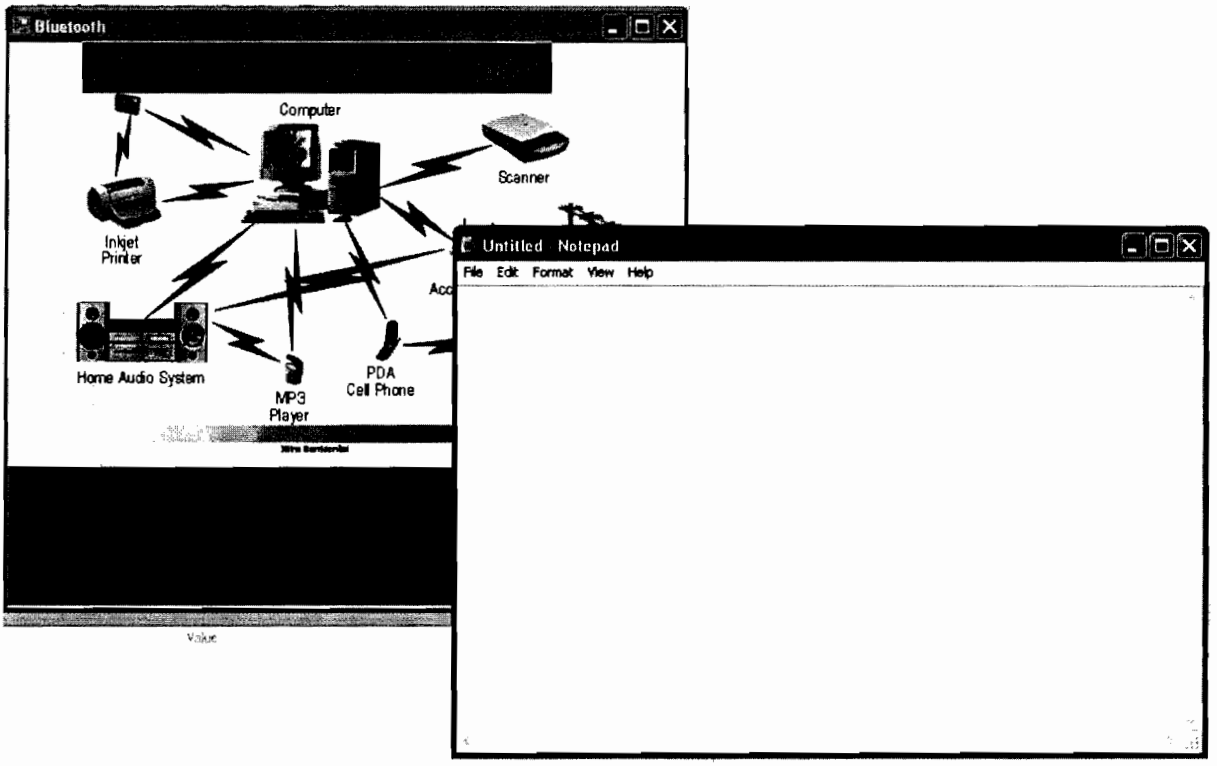

Figure E.4: Action in the server

The client does not allow to creating the file in the directory and delete it only shows the Notepad will opened now.