Accessing PC Using Mobile Device via Bluetooth Technology

This project paper is submitted in partial fulfillment of the requirement for the degree of
Bachelor of Science in Computer Science & Engineering

Course Code: **CIT 402**
Course Title: Project & Thesis

Supervised by:
Hasi Saha
Lecturer
Department of Computer Science & Information Technology
Faculty of Computer Science & Engineering

Submitted By:

Md. Amirul Islam Jiban
Student ID: 0702012
B.Sc. in CSE
Faculty of CSE

Reshad Bin Kibria
Student ID: 0802035
B.Sc. in CSE
Faculty of CSE

Submitted To:
Department of
Computer Science & Information Technology
Hajee Mohammad Danesh Science & Technology University,
Dinajpur, Bangladesh

January, 2013

Faculty of Computer Science & Engineering

Hajee Mohammad Danesh Science & Technology University
Dinajpur-5200, Bangladesh

CERTIFICATE

This is to certify that the project paper entitled **"Accessing PC Using Mobile Device via Bluetooth Technology"** *submitted by* **Reshad Bin Kibria, Student ID: 0802035** *and* **Md. Amirul Islam, Student ID: 0702012** *have been carried out under my supervision. This project has been prepared in partial fulfillment of the requirement for the Degree of Bachelor of Science in Computer Science & Engineering.*

[illegible] 25.01.13
....

Co-Supervisor

Ashis Kumar Mandal

Assistant Professor

Department of CEN

Hasi 27.01.13
....

Supervisor

Hasi Saha

Lecturer

Department of CIT

Hajee Mohammad Danesh Science & Technology University, Dinajpur-5200

January, 2013

DEDICATED TO

OUR RESPECTABLE PARENTS

WHO ARE THE SOURCE OF OUR LIFE IN THIS EARTH

ACKNOWLEDGEMENTS

First of all we acknowledge to almighty ALLAH for completing this project successfully. Then we are grateful to our project supervisor, ***Lecturer Hasi Saha***, and co-supervisor, ***Assistant Prof. Ashis Kumar Mandal***, whose intelligent direction has made the task easier to accomplish. A special thanks to the Chairman of the Department of CIT, ***Assistant Prof. Delowar Hossain***, who has assigned us this project and guide us in the whole project from initial stage to the end that makes us very proud. Without their valuable help and guidance, the completion of this project would have been a difficult task.

ABSTRACT

The objective of our project is to develop software that will perform basic PC controlling operations using Bluetooth technology. We used an IDE (Integrated Development Environment), named NetBeans IDE 7.1.2, which gives us efficient and comprehensive facilities to develop the software. There are two programs we developed for our desired software (HSTU-MobiX), one program is for the mobile which works as the client and the other is server program. The client first discovers only the intended server site program and discovers what services it contains. Then the client program sends some stream data as command to PC's virtual communication port (RFCOMM). The server program reads stream data from the communication port and executes those commands.

TABLE OF CONTENTS

List of Figures

LIST OF TABLES

Chapter 1: Introduction

1.1 Motivation

As the number of Bluetooth products increases each year, it is important to develop applications and services to take full advantage of their potential and capabilities. So we want to develop an application which gives us an inter-connection between the PC and the mobile device. This application helps us to access PC remotely and making fun.

1.1.1 Evaluation of existing systems

We searched for similar software on Internet and listed their features and functionalities, and the tools used to develop those software. Unfortunately whether they are all limited to certain application such as PowerPoint, Windows Media Player, and Windows Explorer or give no security to the user. So we had the opportunity to develop software that gain control to the mouse and the keyboard and come up with jam-packed features and applications.

1.1.2 Candidate system

Inspecting those existing systems, we wanted to develop our system by using the possible features feasible such as mouse controlling, keyboard controlling, applications using and accessing utilities with an enhanced security.

1.2 Objective

Develop software that can be used to access Personal Computer or Notebook using mobile device via Bluetooth Technology to do the following tasks:

- Accessing PC using mouse & keyboard movement
- Explore all the drives and it's subfolders and files
- Play and control songs in Media Players (WMP, Winamp, VLC)
- Control various windows applications e.g., Windows Explorer, Firefox, IE, Power Point from a remote place without the use of Keyboard or mouse
- Accessing Windows RUN Command

1.3 Idea of Our Project

Our idea is very simple. Assume we have two programs, one for the server (Computer) and other for the client (Mobile). And we have a communication protocol (RFCOMM) to communicate the client with the server. Then it's easy for client to send data streams to the server and server program show the output by executes those commands under certain method.

Chapter 2: Bluetooth Technology

2.1 Bluetooth

Figure 1: Bluetooth Technology

Bluetooth is a wireless communication protocol mainly used for short distance and in devices with low power consumption. Because Bluetooth is capable of communicating in an Omni-directional manner of up to 30 feet at 1 Mb/s it is far superior to infrared [5]. Where infrared requires a distance of a few feet or less and requires a direct line of site for transmissions. The Bluetooth core system consists of a radio frequency (RF) transceiver, baseband, and protocol stack. The system offers services that enable the connection of devices and the exchange of a variety of classes of data between these devices. Actually it's a wireless communication protocol that, like HTTP or FTP, operates in client-server architecture. It uses the 2.4 GHz band. If there are multiple peripherals to be connected to a computer using RS-232 or USB, then Bluetooth is the ideal solution to use those devices wirelessly [6].

2.2 The Bluetooth Protocol Stack

The Bluetooth stack and Bluetooth hardware has close relationship. It works as the driver for the Bluetooth hardware. The Bluetooth stack is a controlling agent (it could be software, firmware, hardware, or a combination of all three) that implements the Bluetooth protocol and also allows controlling Bluetooth device programmatically [7].

In order to communicate with the Bluetooth protocol and to control a Bluetooth radio, your computer uses a Bluetooth stack.

Table 1: Layers of the Protocol Stack and their functions [8]

Short Name	Full Name	Description
HCI	Host Controller Interface	The layer that interfaces the host (i.e., the PC) and the controller (the Bluetooth module)
L2CAP	Logical Link Control and Adaptation Protocol	The layer that handles all data transmissions from upper layers
SDP	Service Discovery Protocol	The layer that discovers services on Bluetooth devices in the area
RFCOMM	RFCOMM	The layer that allows you to create a virtual serial port and to stream data
TCS-BIN	Telephony Control Protocol Specification	The layer that allows you to create control signals for audio applications
WAP	Wireless Access Protocol	The adopted protocol that allows you to view content in Wireless Markup Language (WML)
OBEX	Object Exchange	The adopted protocol that allows you to send and receive objects
BNEP	Bluetooth Network Encapsulation Protocol	The layer that encapsulates other protocol data packets into L2CAP packets
HID	Human Interface Device Protocol	The layer that traffics the controls signals and data for input devices like keyboards and mice

We used SDP and RFCOMM layers for the communication between the client and the server. Following are the short descriptions of them.

RFCOMM

RFCOMM is commonly known as the wireless serial port, or the cable replacement protocol [9]. The name is derived from the fact that the serial ports are called COMM1, COMM2, etc. RFCOMM simulates the functionality of a standard serial port. For instance, a Bluetooth-enabled mobile would use the RFCOMM layer to synchronize its data to a Bluetooth-enabled PC as if they were physically connected by a cable.

Service Discovery Protocol (SDP)

A Bluetooth device uses Service Discovery Protocol in order to discover services in the areas. SDP also used to allow devices to discover what services each other support, and what parameters to use to connect to them. For example, when connecting a mobile phone to a Bluetooth headset, SDP will be used to determine which Bluetooth profiles are supported by the headset (headset profile, hands free profile, advanced audio distribution profile, etc.) and the protocol multiplexer settings needed to connect to each of them [9].

2.3 Bluetooth Profiles

To be able to communicate via Bluetooth, devices must not only support the Bluetooth protocol but they must also support a common Bluetooth profile. A Bluetooth profile is a functionality set for Bluetooth devices. Below is a list of some profiles with an explanation of what they do.

- GAP (Generic Access Profile): Device discovery and link management.
- GOEP (Generic Object Exchange Profile): Support files transfer and object push.
- SPP (*Serial Port Profile): Setting up a virtual link between two peer devices.*
- HS (Headset Profile): Defines requirements to support the use of headsets.

2.4 The Java Bluetooth API

To control the Bluetooth device programmatically, J2ME optional package JSR 82 is needed. JSR-82 can only be implemented on the J2ME platform. JSR-82 cannot be implemented on the J2SE because the J2SE does not support the generic connection framework. The JSR-82 actually consists of two independent packages [10]:

1. javax.bluetooth (the 13 classes and interfaces that are needed to perform wireless communication with the Bluetooth protocol)

2. javax.obex (the 8 classes that are needed to send objects between devices, independent of the transport mechanism between them)

2.5 The Basic Components of a Bluetooth Application

The basic components of any Bluetooth application consist of the following items [11]:

- Stack initialization
- Device management
- Device discovery
- Service discovery
- Service registration
- Communication

2.5.1 Stack Initialization

Now before doing anything, stack needs to be initialized. Remember, a Bluetooth stack has direct access to the underlying Bluetooth device. Stack initialization can consist of a number of things, but its main purpose is to get the Bluetooth device ready to start wireless communication. Stack initialization sequences can vary, and it's heavily dependent upon the underlying OS and Bluetooth radio.

2.5.2 Device Management

LocalDevice, RemoteDevice and DeviceClass are the classes in the Java Bluetooth specification that form the Generic Access Profile and allow you to perform device management. These classes allow you to query some statistical information about your own Bluetooth device (LocalDevice) and also some information on the devices in the area (RemoteDevice). The DeviceClass object gives you information about the official class of device (CoD) as defined in the Bluetooth specification.

2.5.3 Device Discovery

The Bluetooth device has no idea of what other Bluetooth devices are in the area. Perhaps there are laptops, desktops, printers, mobile phones, or PDAs in the area. The possibilities are endless. In order to find out, the Bluetooth device will use the device discovery classes that are provided in the Java Bluetooth API.

The two classes needed in order for your Bluetooth device to discover remote Bluetooth devices in the area: DiscoveryAgent and DiscoveryListener.

The method DiscoveryAgent is used to make the Bluetooth device search for other devices in the area. The length of the inquiry is totally dependent upon the implementation of the Java Bluetooth specification. The accessCode can be one of the following DiscoveryAgent constants:

Table 2: Bluetooth Discovery Modes [12]

Access Mode	**Full Name**	**Description**	**Value**
NOT_DISCOVERABLE	Not Discoverable	Don't allow any devices to discover our device	0
GIAC	General/Unlimited Inquiry Access Code	Allow all devices to discover our device	10390323
LIAC	Limited Inquiry Access Code	A temporary access mode that will revert back to a *previous state after* 1 minute	10390272

A reference to a class that implements the DiscoveryListener interface is also passed. When new devices are discovered, event callbacks are passed back to this object. This method will *return true if the device successfully went into discovery mode. The startInquiry() method is the* only way to perform device discovery without blocking the current thread.

2.5.4 Service Discovery

After locating devices in the area, it would be really nice to see what services those devices offer. The service discovery-related classes in the Java Bluetooth specification implement the Service Discovery Application Profile. The Service Discovery Application Profile, in turn, uses the Service Discovery Protocol (SDP) layer in your Bluetooth stack to find services on remote Bluetooth devices.

The following classes are provided in the Java Bluetooth specification for service discovery:

DiscoveryAgent class provides methods to perform device and service discovery. **DiscoveryListener** class allows an application to receive device discovery and service discovery events.The Service Discovery Database (SDDB) is the central repository for all service records. **DataElement** class defines the various data types that a Bluetooth service attribute value may have. The **UUID** class is simply a class that uniquely identifies services in the Bluetooth protocol (UUID stands for Universal Unique Identifier).

Table 3: Common UUID Values for Bluetooth Protocol Layers [13]

Protocol	UUID (Decimal)	UUID (Hexadecimal)
SDP	1	0X0001
RFCOMM	3	0X0003
L2CAP	256	0X0100
HTTP	12	0X000C
OBEX	8	0X0008

Service records in the SDDB as Java objects, then it must convert them into Service Record objects when a client performs a search for services and a match is found.

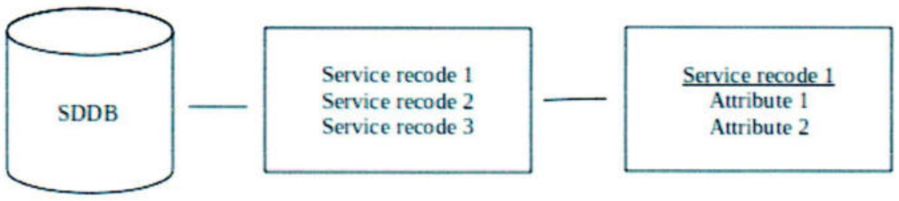


Figure 2: SDDB (Service Discovery Database)

2.5.5 Service Registration

Before a Bluetooth client device can use service discovery on a Bluetooth server device, the server needs to register its services internally. That process is called service registration.

Here's a scenario of what's involved in getting your service registered and stored in the SDDB:

1. Call Connector.open() and cast the resulting connection to a StreamConnectionNotifier object. Connector.open() creates a new ServiceRecord and sets some attributes.
2. Use the LocalDevice object and the StreamConnectionNotifier to obtain the ServiceRecord that was created by the system.
3. Add or modify the attributes in the ServiceRecord (optional).
4. Use the StreamConnectionNotifier to call acceptAndOpen() and wait for Bluetooth clients to discover this service and connect.
5. The system creates a service record in the SDDB. Wait until a client connects. When the server is ready to exit, call close() on the StreamConnectionNotifier.
6. The system removes the service record from the SDDB.

StreamConnectionNotifier and Connector both come from the javax.microedition.io package of J2ME platform.

2.5.6 Communication

As Bluetooth is a communication protocol the official Java Bluetooth API gives three ways to send and receive data. Two of them are: RFCOMM for stream data and L2CAP for packet data. RFCOMM is the protocol layer that the serial port profile uses in order to communicate the client to the server.

Chapter 3: Software Development Life Cycle

The **Systems development life cycle or Software development life cycle (SDLC)** in systems engineering, information systems and software engineering, is a process of creating or altering information systems, and the models and methodologies that people use to develop these systems [14]. We can say that the software development life cycle is the set of activities that analysts and designers carry out to develop and implement an information system, includes preliminary investigation – Recognition of need, Feasibility study, collection of data and determination of requirements, design of system, development of software and system testing and implementation.

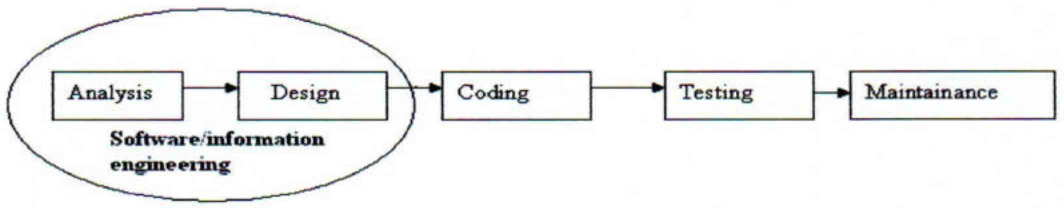


Figure 3: Software Development Life Cycle

3.1 Recognition of Need

Our problem was to develop a server side and a client side program in which client side program can access the server side program's functionalities via Bluetooth.

3.2 Feasibility Study

Whenever a project is initiated, it is necessary to check whether the new system is feasible to develop or install. The main objective is to determine whether the development of the project has a reasonable chance of success.

There were three types of Feasibility Study carried out:

- Economic Feasibility
- Technical Feasibility
- Behavioral Feasibility

3.2.1 Economic Feasibility

In the economic feasibility, the project was considered and cost estimation was done that is how much cost will be appropriately required for overall development of the project and will the project be actually implemented and will the clients find the software to be useful enough to implement for their personal use.

3.2.2 Technical Feasibility

In the technical feasibility, the project was checked and the software and the hardware requirements of the project were pre-calculated. In checking the project for the technical feasibility, we considered that will the software run in the current hardware and software.

3.2.3 Behavioral Feasibility

In this feasibility study, the project was checked and assumed that if the software is actually developed, will the interface provide user friendly enough for a non-computer educated client to work with the software. Will the clients in the organization accept this software?

3.3 Analysis

In order to study, to examine something or to learn what it is made up of, analysis phase is needed. The analysis is about understanding the design environment.

The analysis also consists of identifying software and hardware requirements. So to develop our candidate system we had to use the following technologies:

- NetBeans IDE for developing software primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications and others.
- The Java Development Kit (JDK) for implementation of either one of the Java SE, Java EE or Java ME platforms released by Oracle Corporation.
- The Sun Java Wireless Toolkit (WTK) for developing wireless applications that are based on JavaME's Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP), and designed to run on cell phones, mainstream personal digital assistants, and other small mobile devices.
- Bluecove and Piccolo Java Class Libraries.
- Win32lib.dll and Intelbth.dll files.

Analysis phase is entitled the Data Flow Diagram and the Activity Diagram.

3.3.1 Data Flow Diagram (DFD)

The Data Flow Diagram (DFD) is a graphical representation of the flow of data through an information system. It shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored [15]. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (flowchart). A data-flow diagram can also be used for the visualization of data processing (structured design).

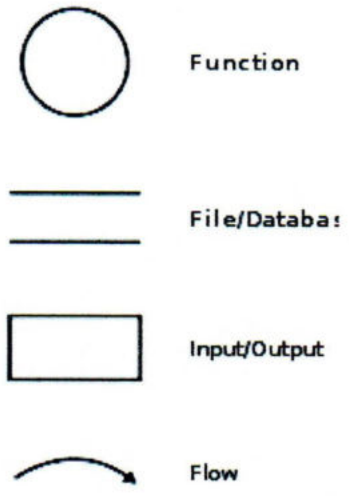


Figure 4: DFD (Data Flow Diagram) – Notation [15]

The **Context-level data flow diagram** shows the interaction between the system and external agents which act as data sources and data sinks [16]. It also shows the entire system as a single process, and gives no clues as to its internal organization. This context-level DFD is next exploded to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 0 DFD shows the overall context of the system and its operating environment and shows the whole system as just one process. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.

The Context-level data flow diagram of our candidate system is given as follows:

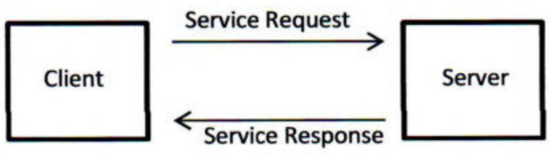


Figure 5: CONTEXT LEVEL DIAGRAM (LEVEL 0)

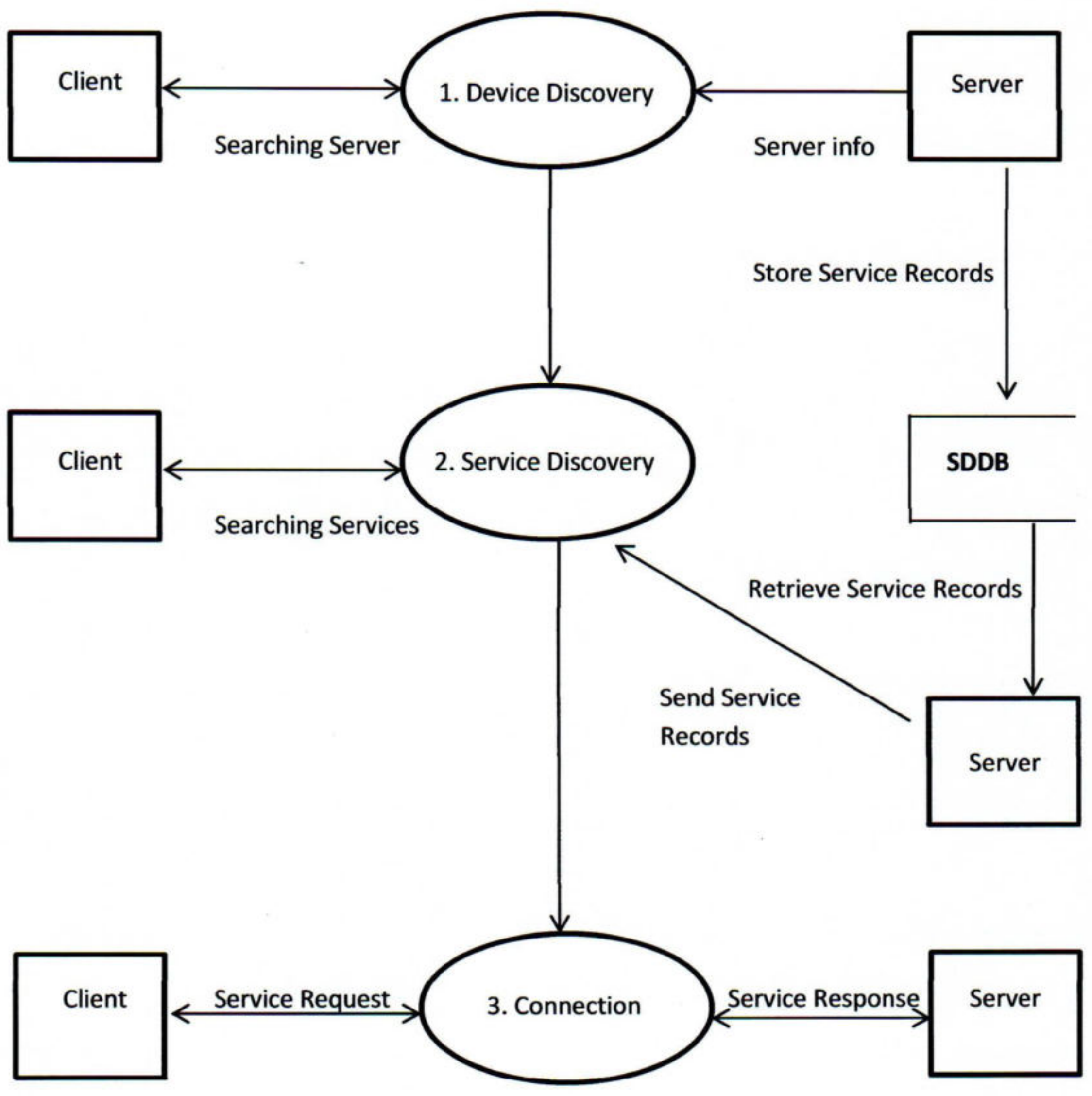


Figure 6: CLIENT-SERVER (LEVEL 1)

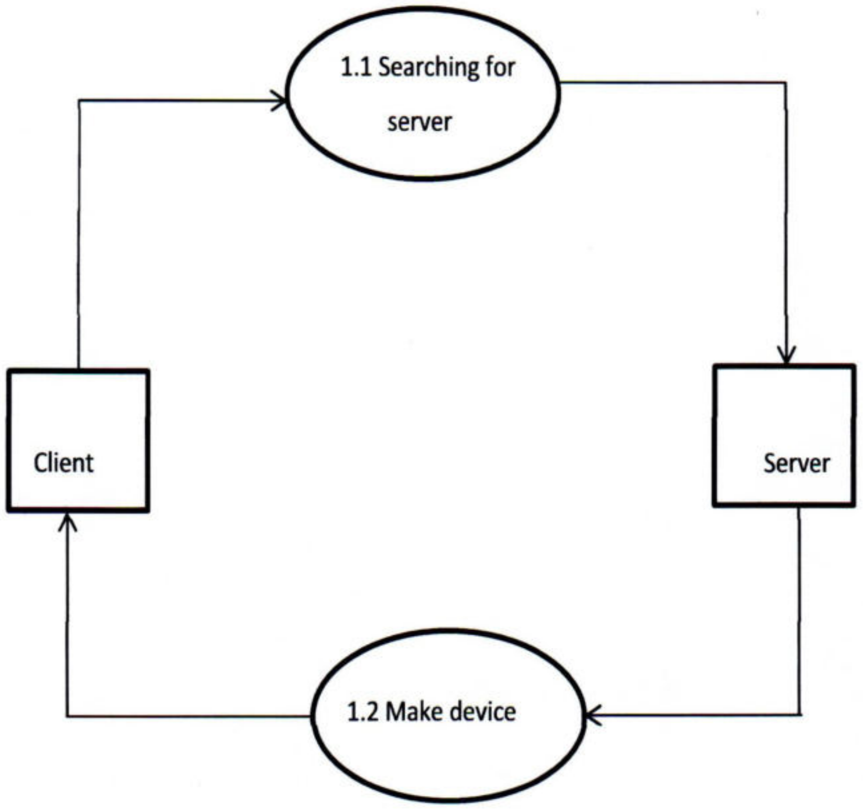


Figure 7: DEVICE DISCOVERY (LEVEL 1)

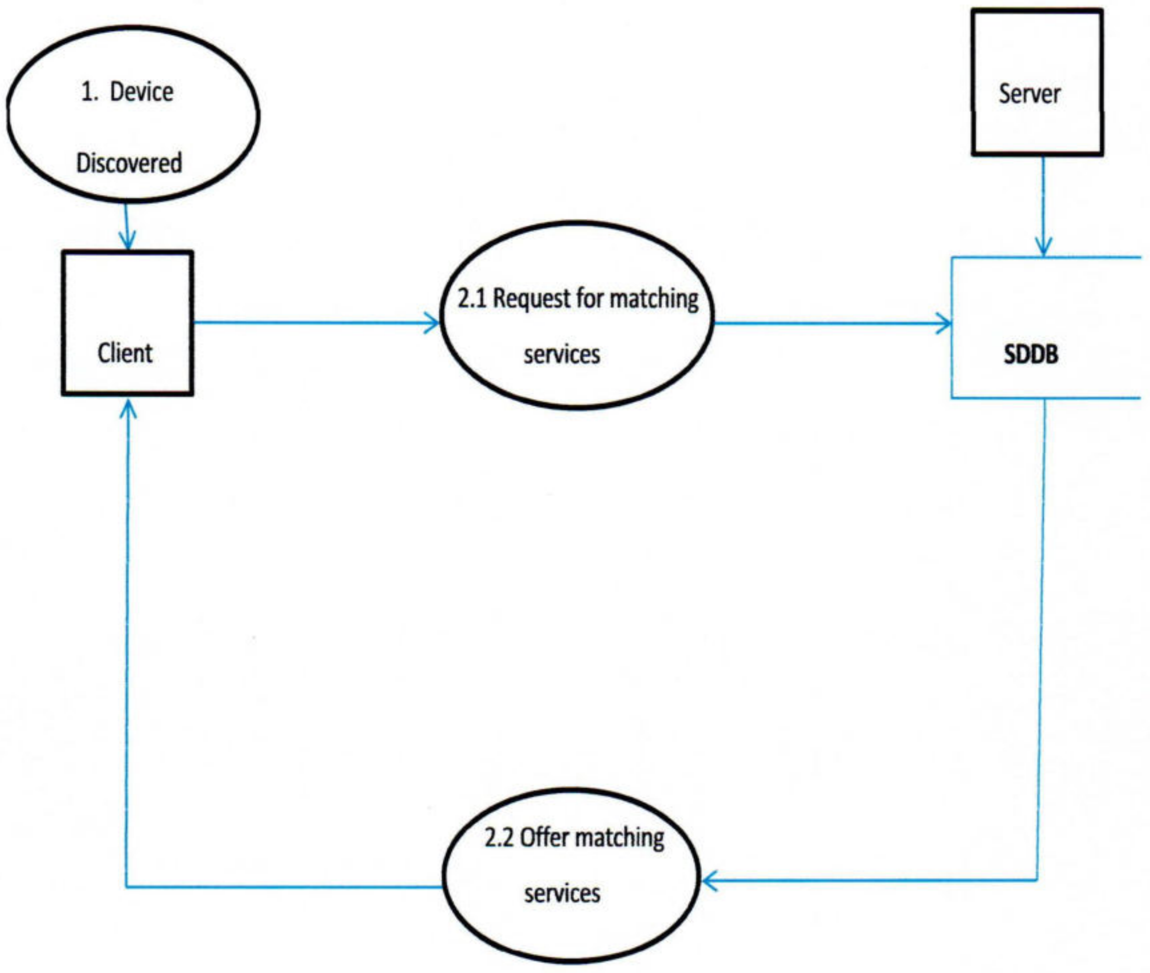

1. Device
Discovered
Server
Client
2.1 Request for matching services
SDDB
2.2 Offer matching services

Figure 8: SERVICE DISCOVERY (LEVEL 1)

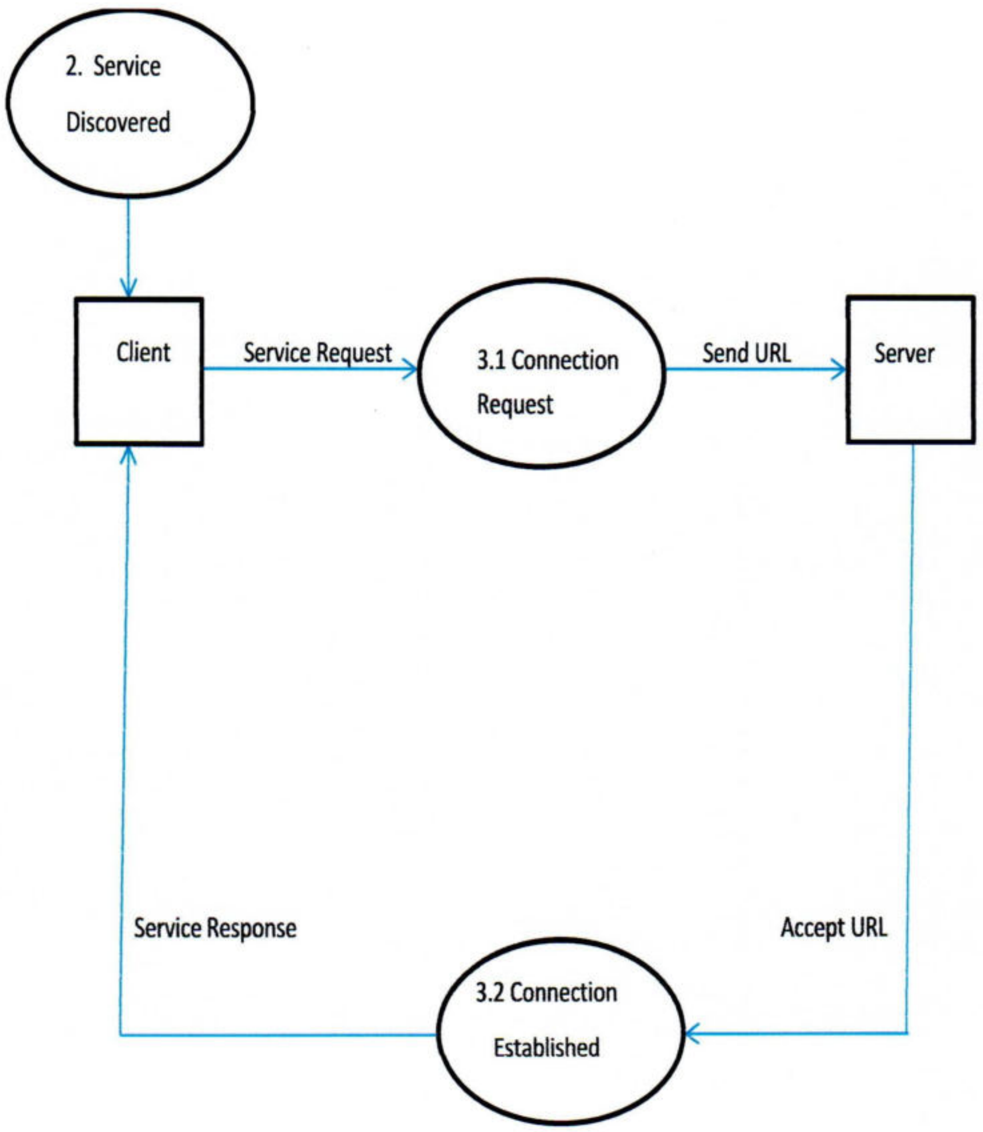


Figure 9: CONNECTION (LEVEL 1)

3.3.2 Activity Diagram

Activity diagram represents the business and operational workflow of a system. An activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in a particular state. An activity diagram talks more about the transitions and activities causing the changes in the object states.

Activity diagrams are typically used for business process modeling. They consist of:

- Initial node
- Activity final node
- Activities

The starting point of the diagram is the initial node, and the activity final node is the ending. An activity diagram can have zero or more activity final nodes. In between activities are represented by rounded rectangles. A typical activity diagram is as shown below.

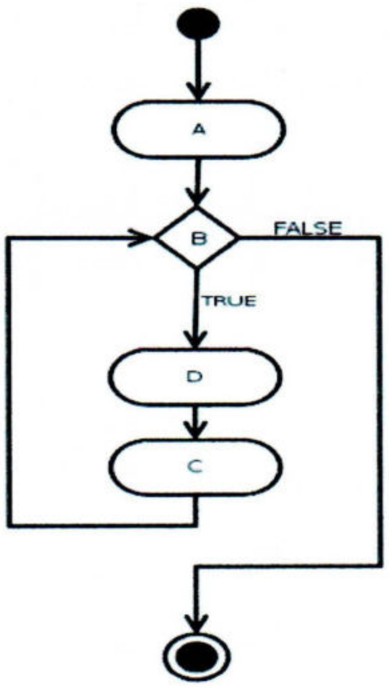


Figure 10: ACTIVITY DIAGRAM

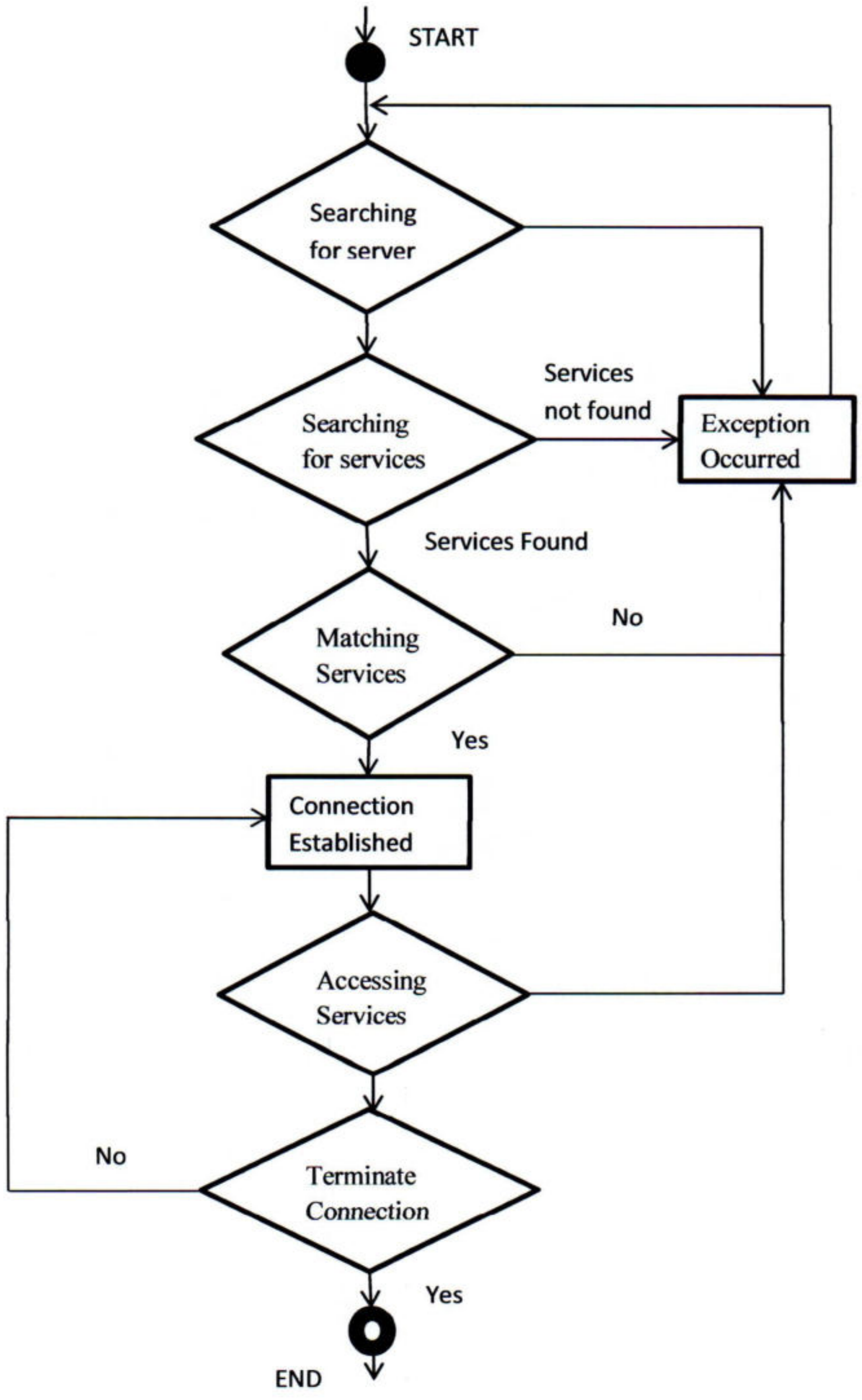


Figure 11: Activity Diagram of the Candidate System

3.4 Design

Design has been described as multi-step process in which representation of data and program structure, interface characteristics and procedural detail are synthesized from information requirements. Design is essentially the bridge between requirement specification and the final solution for satisfying the requirements.

3.4.1 Architectural Design

The architectural design identifies the multiple components and the interaction among them. It provides enough detail on the component to write and detail designs. During this design process, top level designs, all forms, report, data structures, program module and human interfaces based on the information contained in the System Specification document. It also identifies the design criteria and design constraints applicable.

Primary objective of the architectural design is to develop a modular program structure and represent the control relationship between modules. In addition, it melds program structure and data structure defining interfaces that enable data to flow throughout the program.

The Architectural Design of the candidate system was developed in this process.

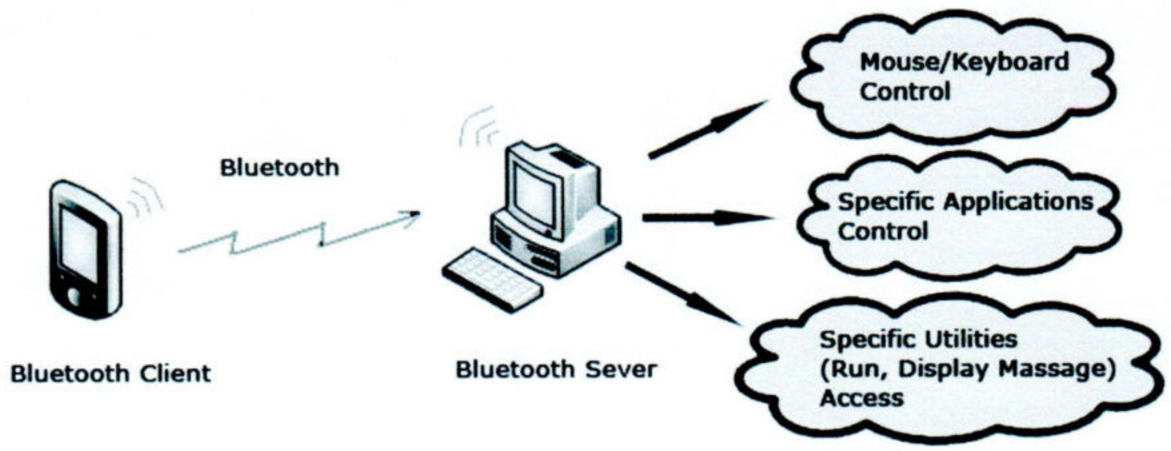

Mouse/Keyboard Control
Bluetooth
Specific Applications Control
Specific Utilities (Run, Display Massage) Access
Bluetooth Client
Bluetooth Sever

Figure 12: Architectural Design of the Candidate System

3.4.2 Functional Design

The Functional design procedures of our candidate system are followed as:

- Server continuously opens for detecting new Bluetooth devices in the Bluetooth range
- Bluetooth client in range turns on its Bluetooth Connection
- Server detects connection and stores service records in the SDDB for service registration
- Client searches in remote area for server
- After detecting the server, the client requests for matched services
- Server provides services to the client if the services match with the client's request
- Client sends a request for services by sending a URL, server accepts the URL, and connection is established between the server and the client
- After successful connection, client can send command to the server and get access to the services the server provides

3.4.3 User Interface

The Interface design focuses on two areas of concern:

- The design interface of server components
- The design interface of client components

The system has following forms for user interface:

- HSTU-MobiX Server
- HSTU-MobiX Client

3.5 Coding and Testing

3.5.1 Coding

The goal of the coding and programming phase is to translate the design of the system produced during the design phase into code in a given programming language which can be executed and which performs specified by the design.

The coding of our client program is developed in J2ME and the server program is developed in J2SE JAVA languages.

3.5.1.1 Client Side Program

We developed a MIDLET for mobile device which will work as the client side program. We gave the name of our client program "HSTU-MobiX" (Client). We used several classes to build the client program.

The core J2ME MIDlet program is the **MobiXClient.java.** A MIDlet is a Java application that runs on a mobile device and uses the Mobile Information Device Profile (MIDP). One or more MIDlets packaged together in a JAR file constitute a MIDlet suite. Software on the mobile device (called the application manager) is responsible for loading, running, and destroying the MIDlet.

The MIDP UI is logically composed of two APIs: high-level and low-level. The high-level API is primarily designed for business applications, and it gives objects like List, TextBox, ChoiceGroup, and DateField. This API includes a high level of abstraction because you can't define the visual appearance (i.e., shape, color, font, etc.) of those components. The low-level API, on the other hand, is designed for applications that need precise placement and control of graphic elements.

For communicating with the server, Bluetooth technology is used. The **Bluetooth.java** class provides the connection with the server. This class used the bluetooth the bluetooth components: Device Discovery and Service Discovery.

Other classes used to develop the HSTU-MobiX client program is dipicted in the following tables.

Table 4: Classes used in HSTU-MobiX client program and their uses

Classes	**Description**
AppList.java	This class displays the applications on the server that can be controlled by the client
CommandsList.java	This class displays the list of available commands for a specified application
CommandsTable.java	This class maps keystrokes to values into an array called hashtable
DevList.java	Displays the available Bluetooth Servers
HelpForm.java	This Class is used for providing help guidelines to the user
MainCanvas.java	Provides the GUI interface of the client
MainList.java	Provides the list of available choices for images or strings available to the client
ModeList.java	Provides the list of available operational modes with GUI
RunCommand.java	*This class is used to run the windows command using a TextBox user* interface
SendMessage.java	This class is used to send a message to the server
SplashScreen.java	SplashScreen class is used for viewing images of exceptions or errors

Service Discovery

Just like Device Discovery, the Service Discovery is done by the DiscoveryAgent class. The method searchServices() is used to search services and when services are found, the JVM will call the servicesDiscovered() method in the object that implemented the DiscoveryListener interface and pass a ServiceRecord object that correspond to the service. With this ServiceRecord it is possible to retrieve the remote devices URL.

Before a Bluetooth client device can do a Service Discovery on a Bluetooth server device, the Bluetooth server has to register its services in internally in the Service Discovery Database (SDDB) [ref]. This is called Service Registration. To do a Service Registration the server calls Connector.open() method and cast the resulting connection to a StreamConnectionNotifier. The Connector.open() method creates a ServiceRecord. This 2 Bluetooth ServiceRecord can be modified by the server. When the server sets itself in communication mode, that is calls the StreamConnectionNotifier.acceptAndOpen() method, and the system automatically creates a ServiceRecord in the SDDB.

Communication

One way to send and receive data in Java Bluetooth is to use the RFCOMM protocol. The Serial Port Profile (SPP) uses this protocol to communicate. There are two ways to start communicating in Java Bluetooth; either as a server or as a client.

After the server has done its service registration and set itself into discoverable mode, it is ready to communicate. To start the communication the server creates a StreamConnectionNotifier object. This object is used to instantiate a StreamConnection object that is used to receive incoming client connections.

After a client has done device and service discovery the client can start the communication. This is done by making a stream connection to the server with the retrieved URL from the service record. When a server accepts a connection from a client it will become the slave and the client will become the master.

3.5.1.2 Server Side Program

The server side program was mainly developed to use in the PC (Personal Computer). For that we used J2SE java languages. The server side program acted as a receiver which received various commands from the client program and took necessary steps to perform different actions. İt contained all the necessary informations to receive commands from the client to contrtol the PC.

The server side program was functioned with different activities. The functions of the server program are discussed as follows:

The Bluetooth Connection

For communicating with the client we used Bluetooth technology. We used **Bluetooth.java** class which provides the connection with the client. We also used our server main class by importing it into the **Bluetooth.java** class. The **Bluetooth.java** class used the bluetooth components: Device Management and Service Registration. This class used different built-in classes (LocalDevice, UUID, StreamConnection etc.) and methods (getLocalDevice(), setDiscoverable(), connector.open()) as well as user-defined methods (listen(), close(), SendData() etc.) to establish the connection between the server and the client. The **Bluetooth.java** class used RFCOMM protocol layer and SDP (Service Discovery Protocol) protocol for the connection.

The Server Program

The main server program was developed in the **MobiXServer.java** class. This class was used to receive the commands from the client program (mobile phone) and provide services, the client needs. For Bluetooth stack we used default Microsoft Windows Bluetooth adapter which was initialized in this class with default commands. The measurements of various components of the GUI (Graphical User Interface) section were also initialized in this class.

Mouse Control

The mouse control functionality of our server program was defined with two classes, **MouseControl.java** and **MouseCursor.java**. The **MouseControl.java** class was used to move the mouse cursor. It started moving the cursor until a boolean variable becomes false. For this we used Robot class. This class also sent the client a buffered screen shot of the area around the mouse. The **MouseCursor.java** class was used in order to invoke the win32lib.dll, which gave the windows restricted permission.

Keyboard Control

The keyboard contol class, **KeyboardControl.java** was used to control the keyboard of the PC. This class was also used the pre-built Robot class for the keyboard movement.

The GUI Interface

The GUI (Graphical User Interface) of the server program was defined with two classes, **MobiXGUI.java** and **About.java**. The **MobiXGUI.java** class was used to design the program's main canvas. We used JMenu, JMenuItem, JTextArea, JLabel, JScrollPane etc. for the designing purpose. The **About.java** class showed the information about us in a JDialog.

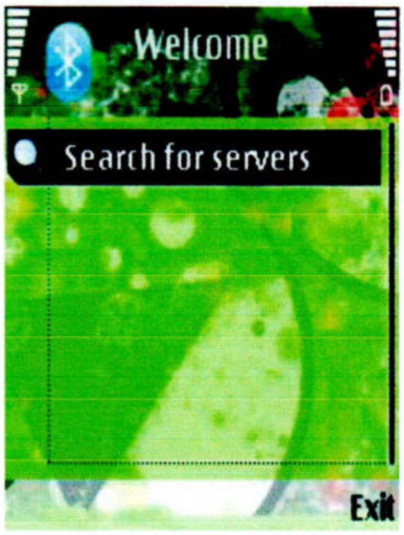
3.5.2 Testing

Testing is a set of activities that can be planned in advance and conducted systematically. Testing is also an individualistic process and number of test varies as the development approaches.

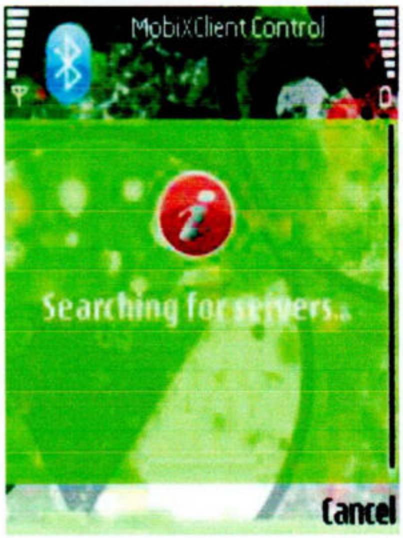
For our developed software, we used different testing approaches on both server and client program to check for the bugs. Some screen shots are provided for both server and client program while testing.

The Client Program Testing

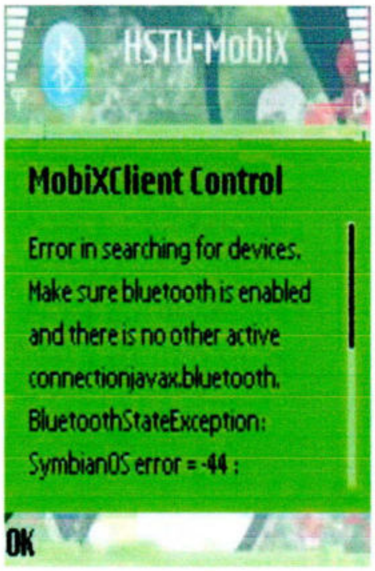
1. The client program's GUI (Graphical User Interface) where "Search for servers" option is visible.



2. The client is searching for the server.



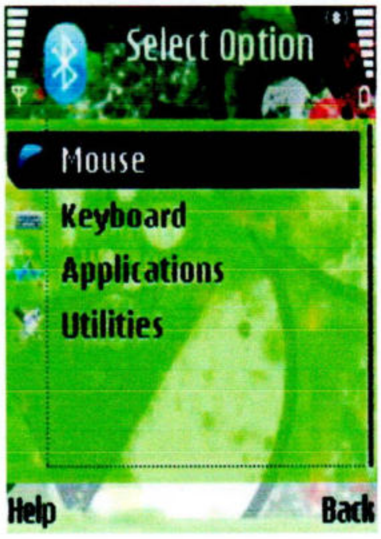
3. If there is any exception in connection, the client program shows about the exception.



MobiXClient Control

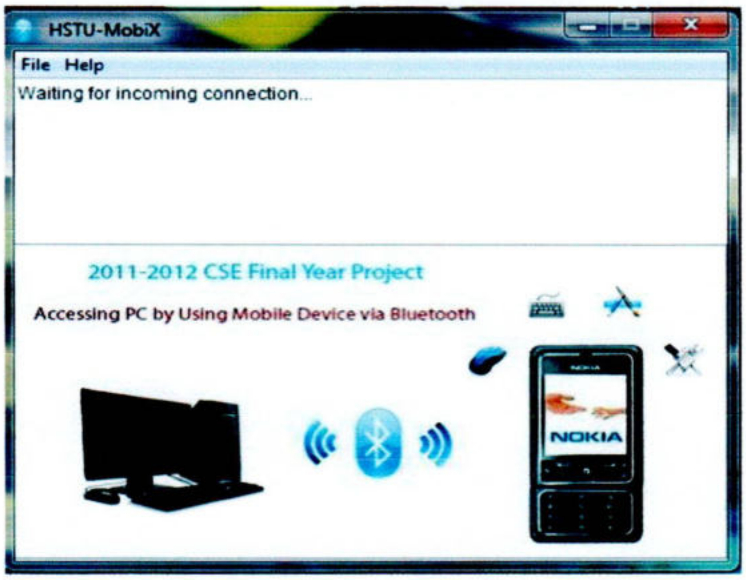
```
Error in searching for devices.
Make sure bluetooth is enabled
and there is no other active
connectionjavax.bluetooth.
BluetoothStateException:
SymbianOS error = -44 :
```

4. When the connection is established the client's GUI shows the options of the services.



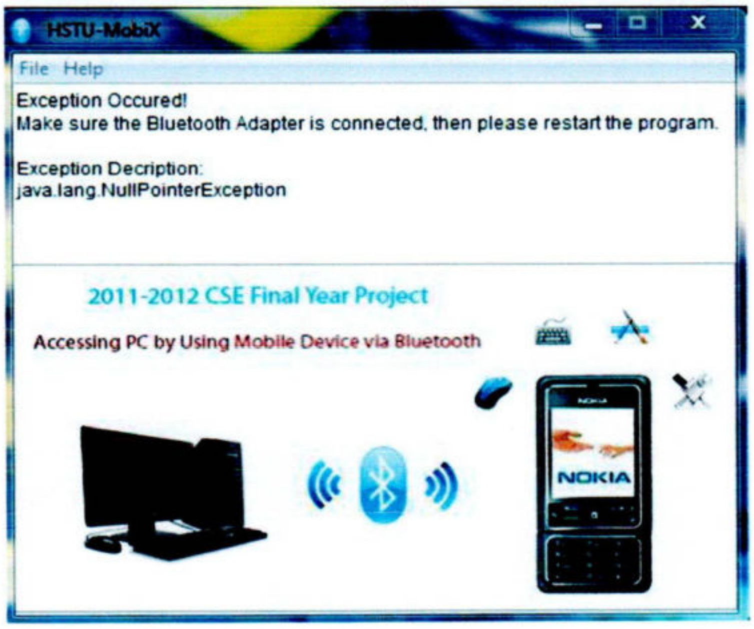
The Server Program Testing

1. The server program's GUI (Graphical User Interface) where shows that the server is waiting for connection to be established with the client.


HSTU-MobiX
File Help
2011-2012 CSE Final Year Project
Accessing PC by Using Mobile Device via Bluetooth
NOKIA

```
Waiting for incoming connection...
```

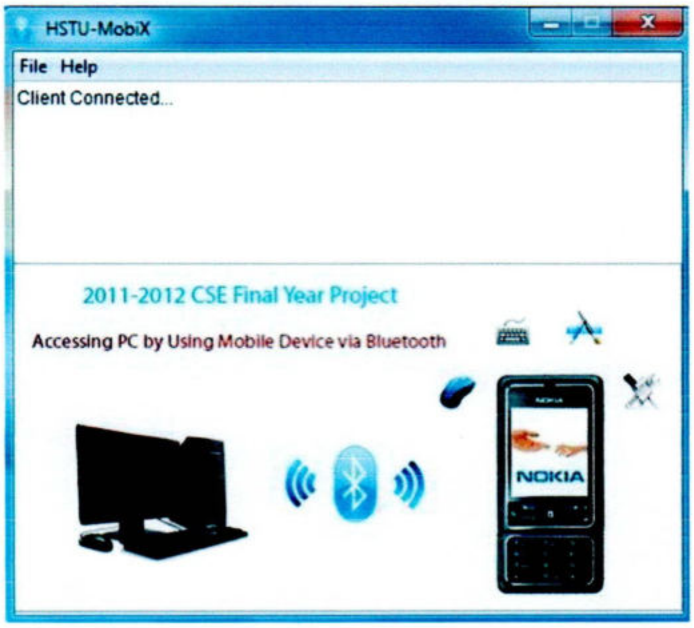
2. When there is an exception occurred, the server shows the message about the exception.


HSTU-MobiX
File Help
2011-2012 CSE Final Year Project
Accessing PC by Using Mobile Device via Bluetooth
NOKIA

```
Exception Occured!
Make sure the Bluetooth Adapter is connected, then please restart the program.

Exception Decription:
java.lang.NullPointerException
```

3. When the connection is established between the server and the client, then the server shows that the client is connected.


HSTU-MobiX
File Help
2011-2012 CSE Final Year Project
Accessing PC by Using Mobile Device via Bluetooth
NOKIA

Client Connected...

3.6 Installation

3.6.1 Minimum Requirements

Server Side (Computer)

- Windows XP SP1 or greater
- Java SE 1.6 or greater
- Bluetooth adapter compatible with Microsoft Bluetooth stuck

Client Side (Java Supported Mobile Phone)

- A mobile phone with support of the Bluetooth API - JSR 82 (no need to support OBEX)

3.6.2 Installation Process

There are two parts of our software. One is developed for the mobile phone (client sided) which must be installed in appropriate mobile phone. To do so,

1. Transfer the **HSTU-MobiX_Client.JAR** and **HSTU-MobiX_Client.JAD** files to the mobile phone provided with the software package. Use Bluetooth Serial port enable mobile.
2. Install it.

The other part is developed for the personal computer (server program) which processes the upcoming command and takes actions. This software is bundled with **JRE-1.6** and does not require pre-installation of **JAVA**. It is also required to attach a Bluetooth receiver to your computer if it is not capable of receiving Bluetooth signal. It is suggested to use a Bluetooth dongle.

Now,

1. Install driver for your Bluetooth receiver device.
2. Install the Receiver software. To do that just double click the "**HSTU-MobiX.exe**" file and the software will be automatically installed.

3.7 Maintenance

Initially the server must be started.

- ➢ Make sure you have connected your Bluetooth adapter
- ➢ Make sure you have enabled your PC to be discoverable. This can be done by changing the appropriate setting in the Bluetooth settings which can be found in the control panel
- ➢ Run the HSTU-MobiX.exe file which is inside the folder named HSTU-MobiX

The client on the phone has to be started.

- ➢ Turn on your Bluetooth on your mobile phone
- ➢ Make sure your phone is not connected to any other device
- ➢ Run the HSTU-MobiX application
- ➢ Press the search for devices option or used the cached computer name - which is the last device used by the application.
- ➢ If you have chosen the search for devices option your PC should be listed in the devices found list. If not, check that your PC is discoverable and repeat the search again
- ➢ Press select. You will now notice on the server side that the message "Client connected" has been appeared. A warning message may appear in your mobile phone asking for permission to use the Bluetooth connection
- ➢ After connecting a new list appears in your mobile phone where you can select among 4 options. The mouse option where you control the mouse cursor by pressing the numeric

keys, the keyboard option where you can control some important keyboard keys by using the numeric keys and the application option where you can send some commands to some very common applications.

Note that in the application option in order to execute the command successfully the desired application should be in focus on the server PC. Which numeric key corresponds to each action can be found in the help menu of each option, e.g., the utility option where you can find various utilities.

Chapter 4: Conclusion and Discussion

This software is very much user-friendly. To use it one needs just to press buttons to control various PC operations. User do not have to know the Bluetooth address of the server PC. It can discover any Bluetooth device and the services around it. This software is applicable for all versions of WINDOWS operating system. As maximum computer users feel comfort to use this operating system. We are looking forward to improve our software to make it truly platform independent and to implement the software in two way communication.

4.1 Limitations

Though we have developed our software using the platform independent programming language Java but our software is not totally platform independent at all. Our software can only read data from the serial port. It can generate signal from the PC. So we can perform two way operations. Our software also can't initialize the Bluetooth stack for the devices. It needs a matching driver for the devices. Some major connections issues persist which make the software slow in connection. Moreover our software cannot provide the security in communication.

4.2 Future Plans

- Use the Authentication and Encrypted protocols for enhanced security
- Adding voice commands for the operations
- Accessing the internet via PC

Reference List

[1] Paul Deitel, Harvey Deitel, "Java how to program". New Jersey: Pearson Education Inc., Eighth Edition.

[2] James Keogh, "J2ME: The Complete Reference". New York: McGraw-Hill/Osborne, 2003.

[3] Bruce Hopkins, Ranjith Antony, "Bluetooth for Java". Apress, 2003.

[4] Qusay Mahmoud, "Learning Wireless Java 2011". O'Reilly, 2001.

[5] "What is Bluetooth", https://sites.google.com/site/wwwj2memantra/bluetooth

[6] "Bluetooth", Bruce Hopkins, Ranjith Antony, "Bluetooth for Java". Apress, 2003, Page 10-11.

[7] "The Bluetooth Protocol Stack", Bruce Hopkins, Ranjith Antony, "Bluetooth for Java". Apress, 2003, Page 25.

[8] "Layers of the Protocol Stack", Bruce Hopkins, Ranjith Antony, "Bluetooth for Java". Apress, 2003, Page 27.

[9] "RFCOMM, SDP", Bruce Hopkins, Ranjith Antony, "Bluetooth for Java". Apress, 2003, Page 28.

[10] "The Java Bluetooth API", Bruce Hopkins, Ranjith Antony, "Bluetooth for Java". Apress, 2003, Page 47.

[11] "The Basic Components of a Bluetooth Application", Bruce Hopkins, Ranjith Antony, "Bluetooth for Java". Apress, 2003, Page 47.

[12] "Bluetooth Discovery Modes", Bruce Hopkins, Ranjith Antony, "Bluetooth for Java". Apress, 2003, Page 51.

[13] "Common UUID Values for Bluetooth Protocol Layers", Bruce Hopkins, Ranjith Antony, "Bluetooth for Java". Apress, 2003, Page 57.

[14] "Systems development life cycle (SDLC)",
http://en.wikipedia.org/wiki/Systems_development_life-cycle

[15] "Data Flow Diagram (DFD)", Elias M. Awad, "Systems Analysis and Design". New Delhi: Galgotia, Page 171.

[16] "Context-level data flow diagram", Wikipedia.

Websites:

www.en.wikipedia.org

www.google.com

www.code.google.com/p/letthephonedecide

www.bluecove.org

www.miniware.net

www.docs.oracle.com