**PC CONTROL USING ANDROID MOBILE PHONE THROUGH BLUETOOTH**

**A PROJECT REPORT**

***Submitted By***

**SIVASANKAR SABAPATHY**

***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**SONA COLLEGE OF TECHNOLOGY**

**ANNA UNIVERSITY: CHENNAI 600 025**

**APRIL 2012**

**PC CONTROL USING ANDROID MOBILE PHONE THROUGH BLUETOOTH**

**A PROJECT REPORT**

***Submitted By***

**SIVASANKAR SABAPATHY**

***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**SONA COLLEGE OF TECHNOLOGY**

**ANNA UNIVERSITY: CHENNAI 600 025**

**APRIL 2012**

**ANNA UNIVERSITY: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report **“PC CONTROL USING MOBILE PHONE THROUGH BLUETOOTH”** is the bonafide work of **“Sivasankar Sabapathy”** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**DR. B. SATHIYABHAMA, MRS. R. SUBHASHINI,**

**HEAD OF THE DEPARTMENT SUPERVISOR**

Assistant Professor (Senior Grade)

Department of Computer Science and Department of Computer Science and

Engineering, Engineering,

Sona College of Technology, Sona College of Technology,

Thiagarajar Polytechnic College Road, Thiagarajar Polytechnic College Road,

Salem – 636 005 Salem – 636 005

Date of Viva-Voce Examination: 16-04-2012 – 18-04-2012

Internal Examiner External Examiner

**TABLE OF CONTENTS**

**CHAPTER TITLE PAGE**

**ABSTRACT** v

**LIST OF TABLES** vi

**LIST OF FIGURES** vii

**LIST OF ABBREVIATIONS** viii

**1. INTRODUCTION 1**

1.1 OBJECTIVE 2

**2. SYSTEM ANALYSIS 3**  2.1 EXISTING SYSTEM 3

2.2 DRAWBACKS 3

2.3 PROPOSED SYSTEM 4

**3. SYSTEM SPECIFICATION 5**

3.1 PC HARDWARE REQUIREMENTS 5

3.2 MOBILE PHONE REQUIREMENTS 5

3.3 SOFTWARE REQUIREMENTS 6

**4. SOFTWARE DESCRIPTION** **7**

4.1 ANDROID 7

4.1.1 Features 7

4.1.2 Android Architecture 9

4.1.2.1 Applications 9

4.1.2.2 Application Framework 10

4.1.2.3 Libraries 10

4.1.2.4 Android Runtime 11

4.1.2.5 Linux Kernels 11

4.1.3 Application Fundamentals 12

4.1.4 Application Components 12

4.1.5 Unique Aspects of Android OS 14

4.1.6 Android Users 14

4.2 JAVA 15

4.2.1 Principles 15

4.2.2 Versions 16

4.2.3 Java Platform 16

4.2.4 Automatic Memory Management 17

4.2.5 Class Libraries 18

4.2.6 Editions 19

4.3 BLUECOVE LIBRARY 21

4.4 ECLIPSE IDE 24

4.4.1 History 25

4.4.2 Licensing 25

4.4.3 Releases 26

4.4.4 Architecture 26

4.4.5 Rich Client Platform 28

**5. SYSTEM IMPLEMENTATION 29**

5.1 OVERVIEW OF THE PROJECT 29

5.2 MODULE DESCRIPTION 29

5.3.1 PC Server Module 29

5.3.2 Mobile Client Module 30

5.3.2.1 Mouse Module 30

5.3.2.2 Keyboard Module 30

5.3.2.3 Shortcuts Module 31

5.3.2.4 Joystick Module 31

**6. SYSTEM TESTING 32**

**7. CONCLUSION 34**

7.1 FUTURE WORKS 34

**APPENDIX 1 35**

**APPENDIX 2 47**

**REFERENCES 54**

## Abstract

With the advent of touch screen smart phones, human lives have never been the same. Nowadays, smart phones have become powerful tools that can connect to the internet using either second Generation (2G) or third Generation (3G) network to receive information, use the Global Positioning System (GPS) satellites to get directions and use Bluetooth to transfer files. Bluetooth 3.0 is the most widely installed module for cell phones and Personal Computers (PC) today. Home Theatre Personal Computer (HTPC) is slowly in the rise. HTPC are PCs that are connected to a Television (TV) or projector as a display. Many laptop users have started connecting their laptops to a TV.

The proposed system ensures Bluetooth control of the PC using a mobile phone using the Android Operating System. This system avoids the use of 3G and Wireless Fidelity (Wi-Fi). It will emulate a mouse, a keyboard and a simple joystick. It allows us to control the computer within a 6 meter range.

## List of Tables

**Table No. Table Name Page No.**

4.1 Major Releases of Java 16

4.2 Different Releases of Eclipse IDE 26

## List of Figures

**Figure No. Figure Name Page No.**

4.1 Android Architecture 9

4.2 Pie Chart of all Android Smartphone 14

4.3 Java Component Technologies 20

4.4 BlueCove Architecture 23

## List of Abbreviations

1. OS Operating System
2. 2G Second Generation Network
3. 3G Third Generation Network
4. PC Personal Computer
5. HTPC Home Theatre Personal Computer
6. TV Television
7. Wi-Fi Wireless Fidelity
8. Froyo Android OS Version 2.2
9. App Application
10. HTTP Hyper Text Transfer Protocol
11. iOS iPhone OS
12. IDE Integrated Development Environment
13. SDK Software Development Kit
14. API Application Programmer Interface
15. GUI Graphical User Interface

**Chapter 1**

# Introduction

Mobile Application Development is the process by which application software is developed for small low-powered devices such as personal digital assistants or mobile phones. These apps are either pre-installed on the device during manufacture, downloaded by customers from various mobile software distribution platforms, or web applications delivered over HTTP which use server-side or client-side processing to provide an “application-like” experience within a Web browser. There are many mobile OS in the market today. Some include Android, iOS, BlackBerry, Symbian OS, Bada, and Windows Mobile. With today’s mobiles phones getting smarter and containing more features, we can use them to do almost anything.

A HTPC or Media Centre appliance is a convergence device that combines some or all capabilities of a PC with a software application that supports videos, photo, music playback, and video recording. An HTPC and other convergence devices integrate components of a home theatre into a unit co-located with a home entertainment system. An HTPC system typically has a remote control and the software interface normally has a 10-foot user interface design so that it can be comfortably viewed at typical television viewing distances. An HTPC can be purchased pre-configured with the required hardware and software needed to add video programming or music to the PC.

## 1.1 Objective

With the rise of HTPC and laptops connected to TVs, a problem occurs in which to control these PCs from a viewable distance. The existing system helps to controls both with some inconveniences and problems. The goal of the project is to provide a simple app to help control the PC. It allows a connection using Bluetooth, which is very simple to use. Any android smartphone, with an OS Version of 2.2 or above, can use this application to run on a PC. It will mimic a keyboard, a mouse and a simple joystick. It also provides some generic PowerPoint and media player shortcuts.

**Chapter 2**

# System Analysis

## 2.1 Existing System

The natural way to control would be to use a wireless keyboard and mouse. There are alsosome applications in android mobile phones. Some of the existing applications are JM2PC, GMOTE, RemoteDroid, and Wi-Fi Mouse. It provides a remote control your desktop or laptop computer. Some apps can access and edit your files, start and stop applications, control the mouse, keyboard and other devices. RemoteDroid turns your phone into a wireless keyboard and mouse with touchpad, using your own wireless network. It is an app for phone and tablets running on the Android OS.

## 2.2 Drawbacks

The drawbacks of the existing systems are identified as:

* Very bulky and uncomfortable to use external devices
* Very complex to port to android mobile phones
* Not implemented well for touch screen phones
* Battery is drained quickly due to Wi-Fi
* Inane data charges due to 3G
* High complexity in setting up a Wi-Fi network

## 2.3 Proposed System

The proposed system will control a PC using Bluetooth. It makes use of an Android phone. It will emulate a keyboard, a mouse and a simple joystick. The system will reduce the use of Wi-Fi and 3G. The user can connect to the computer in simple and fast connection via Bluetooth. As the emulation of a peripheral occurs in the phone, the app sends the commands via Bluetooth to the PC. In the PC, the server will receive the information and convert to appropriate signals to be executed in the PC.

The proposed system overcomes the problems in the existing system and addresses the following:

* It will connect the PC and Mobile phone using Bluetooth
* It provides an easy interface to use the emulation
* It reduces battery usage
* It reduces data charges
* It is simple to install and run on the computer

**Chapter 3**

# System Specification

## 3.1 PC HARDWARE REQUIREMENTS

PROCESSOR : Intel Core 2 Duo CPU

SPEED : 2.00 GHz

RAM : 1 GB

HARD DISK : 30 GB

KEYBOARD : Standard Keyboard

MOUSE : 2 or 3 Button Mouse

OTHER : Bluetooth 2.1

## 3.2 MOBILE PHONE REQUIREMENTS

PROCESSOR : Qualcomm

SPEED : 600 MHz

OS : Froyo

MEMORY : 256 MB

DISPLAY TYPE : Capacitive Touchscreen

OTHER : Bluetooth 2.1

## 3.3 SOFTWARE REQUIREMENTS

OPERATING SYSTEM : Windows XP or above

FRONT END : Eclipse

LANGUAGE : JAVA

TOOLKITS : JAVA SDK, Android SDK

**Chapter 4**

# Software Description

## 4.1 Android

Android is a software stack for mobile devices that include an operating system (OS), a middleware and key applications. The Android Software Development Kit (SDK) provides tools and Application Programming Interface (API) necessary to develop applications (apps) for the Android platform using Java programming language.

### 4.1.1 Features

The various features of Android are:

* **Application Framework**
  + enabling the reuse and replacement of components
* **Dalvik Virtual Machine (DVM)**
  + a Virtual Machine (VM) optimized for mobile devices
* **Integrated Browsers**
  + based on open source WebKit engine
* **Optimized Graphics**
  + Custom 2D graphics library
  + 3D graphics based on OpenGL ES 1.0 specification
* **SQLite**
  + Structured data storage
* **Media Support**
  + For common multimedia formats
    - Audio (MP3, AAC, AMR)
    - Video (MPEG4, H.264)
    - Image (JPG, PNG, GIF)
* **Global System for Mobile Communication (GSM) Telephony**
* **Bluetooth, Enhanced Data rate for GSM Evolution (EDGE), 3G and Wi-Fi**
  + Hardware dependent
* **Camera, Global Positioning System (GPS), Compass, Accelerometer**
  + Hardware dependent
* **Rich Development Environment**
  + Device emulator
  + Tools for debugging
  + Memory and performance profiling
  + Plugin for Eclipse Integrated Development Environment (IDE)

### 4.1.2 Android Architecture

The following diagram shows the major components for Android OS.

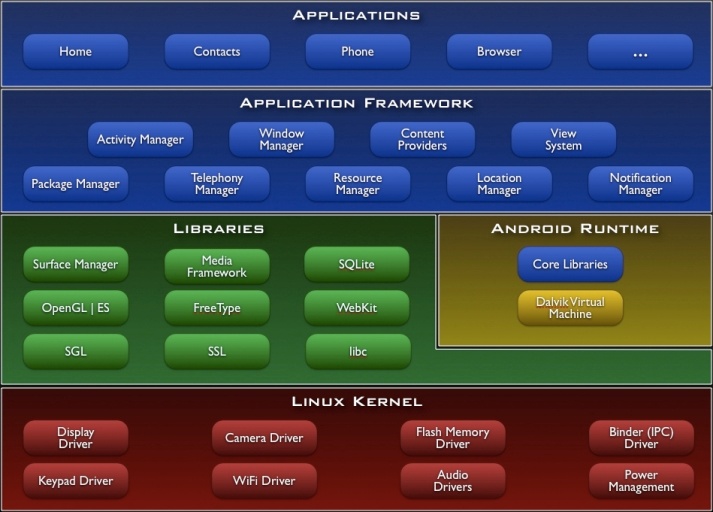


Figure 4.1: Android Architecture

#### 4.1.2.1 Applications

Android ships with a set of core applications written in Java programming language including

* Email client
* Short Messaging Service (SMS)
* Calendar
* Maps
* Browser
* Contacts
* And others

#### 4.1.2.2 Application Framework

The Application Framework is an open development platform. Developers are free to take advantage of device hardware, location information, run background apps, add status information to the notification bar, and much more.

Developers are given full access to same framework that is being used by core apps. The framework simplifies the reuse of components i.e. any app publish its capabilities and any other app can make use of the capabilities if it has the necessary security and authentication. The underlying set of services and systems in Android OS include:

* **A rich and extensible set of Views**
* **Content Providers**
* **Resource Manager**
* **Notification Manager**
* **Activity Manager**

#### 4.1.2.3 Libraries

Android OS include a set of C/C++ libraries used by various components. Some core libraries are:

* **System C Library**
* **Media Libraries**
* **Surface Manager**
* **LibWebCore**
* **SGL**
* **3D Library**
* **FreeType**
* **SQLite**

#### 4.1.2.4 Android Runtime

The Android Runtime includes a set of core libraries provided for each application. Every app runs its own process, with its own instance of DVM. DVM has been rewritten to efficiently process multiple VMs.

DVM relies on the Linux kernel for underlying functionality such as threading and low-level memory management

#### 4.1.2.5 Linux Kernels

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

#### 4.1.3 Application Fundamentals

Android apps are written in Java. The Android SDK compiles the code, data and resources into an **Android package,** an archive file with an .apk suffix. The Android OS has some interesting features. They are:

* **The Android OS is a multi-user Linux where each app is a different user**
* **The system assigns a unique Identification (ID) to each app between the system and the app**
* **Each app has its own VM thus running in isolation from other apps**
* **Each app runs its own Linux process**
* **Android starts a process when an app or its component is needed and shuts down the process when it is no longer needed or system must recover memory from other apps**
* **Android OS implements the principle of least privilege.**
* **Each app, by default, has access to only the components needed and nothing else**
* **However, there are two ways to share data between apps**
  + **The two apps share the same unique Linux User ID (UID)**
  + **The app can request the system for permission to access the components**

#### 4.1.4 ****Application Components****

The application components are the building blocks of an Android app. Each component is a different point through which the system can access the app or its components. There are four types of components. They are:

1. **Activities**
   1. An activity is a single screen of User Interface (UI)
   2. All activities can work together to form a cohesive unit
   3. Each activity is independent of each other
   4. A different app can start an activity
2. **Services**
   1. Services run in background to perform long running process or perform work for a remote process
   2. There is no UI for a service
   3. Another component can start a service and may or may not bind it to the component, depending on the file permissions
3. **Content Providers**
   1. Content providers manages a shared set of app data
   2. There are different ways to store data. They are
      1. File system
      2. SQLite Database
      3. Web
      4. Other persistent storage that can be accessed
   3. Through the content provider, other apps can query or modify the data
   4. It is useful for reading or writing data that is private or confidential to the app
4. **Broadcast Receivers**
   1. Broadcast receivers responds to system wide broadcast announcements
   2. The broadcast originates from the system
   3. Apps can initiate a broadcast
   4. There is no display UI, but it can create status alerts in the notification bar

#### 4.1.5 Unique Aspects of Android OS

Any app can start another app or its component. It does not need to incorporate or link code of component. It just needs to call the activity. Once the activity is performed, the data required is returned back to the app. There is no main method for any app. When a system starts a component, it starts a process for the app and initiates the necessary classes needed for the component.

Since file permissions restrict access to other apps, apps cannot directly access the component. However the Android system can access the component. Thus, the app sends a message to the system with intent to use a component and the system will activate the specified component.

#### 4.1.6 Android Users

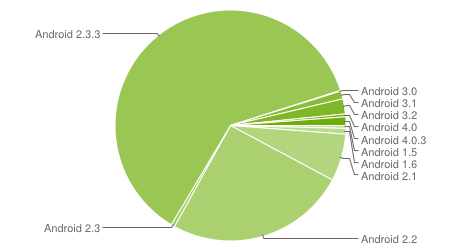


Figure 4.2 Pie Chart of all Android Smartphone accessing Google Play Store

## 4.2 JAVA

Java is a programming language originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. Java is currently one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users.

### 4.2.1 Principles

There were five primary goals in the creation of the Java language:

* It should be "simple, object-oriented and familiar"
* It should be "robust and secure"
* It should be "architecture-neutral and portable"
* It should execute with "high performance"
* It should be "interpreted, threaded, and dynamic"

### 4.2.2 Versions

|  |  |  |
| --- | --- | --- |
| No. | Version | Date |
| 1. | JDK 1.0 | January 23, 1996 |
| 2. | JDK 1.1 | February 19, 1997 |
| 3. | J2SE 1.2 | December 8, 1998 |
| 4. | J2SE 1.3 | May 8, 2000 |
| 5. | J2SE 1.4 | February 6, 2002 |
| 6. | J2SE 5.0 | September 30, 2004 |
| 7. | Java SE 6.0 | December 11, 2006 |
| 8. | Java SE 7.0 | July 28, 2011 |

Table 4.1 Major releases of Java

### 4.2.3 Java Platform

One characteristic of Java is portability, which means that computer programs written in the Java language must run similarly on any hardware/operating-system platform. This is achieved by compiling the Java language code to an intermediate representation called Java byte code, instead of directly to platform-specific machine code. Java byte code instructions are analogous to machine code, but are intended to be interpreted by a virtual machine (VM) written specifically for the host hardware. End-users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a Web browser for Java applets.

Standardized libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

A major benefit of using byte code is porting. However, the overhead of interpretation means that interpreted programs almost always run more slowly than programs compiled to a native executable would. Just-in-Time (JIT) compilers were introduced from an early stage that compilesbyte codes to machine code during runtime.

### 4.2.4 Automatic Memory Management

Java uses an automatic garbage collector to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the unreachable memory becomes eligible to be freed automatically by the garbage collector. Something similar to a memory leak may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a non-existent object are called, a "null pointer exception" is thrown.

One of the ideas behind Java's automatic memory management model is that programmers can be spared the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the stack, or explicitly allocated and freed from the heap. In the latter case the responsibility of managing memory resides with the programmer. If the program does not free an object, a memory leak occurs. If the program attempts to access or free memory that has already been freed, the result is undefined and difficult to predict, and the program is likely to become unstable and/or crash. This can be partially remedied by the use of smart pointers, but these add overhead and complexity.

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

Java does not support C/C++ style pointer arithmetic, where object addresses and unsigned integers (usually long integers) can be used interchangeably. This allows the garbage collector to relocate referenced objects and ensures type safety and security.

Java contains multiple types of garbage collectors. By default, HotSpot uses the Concurrent Mark Sweep collector, also known as the CMS Garbage Collector. However, there are also several other garbage collectors that can be used to manage the Heap. For 90% of applications in Java, the CMS Garbage Collector is good enough.

### 4.2.5 Class Libraries

* The Java Class Library is the compiled byte codes of source code developed by the JRE implementer to support application development in Java. Examples of these libraries are:
  + The core libraries, which include:
    - Collection libraries that implement data structures
    - XML Processing libraries
    - Security
    - Internationalization and localization libraries
  + The integration libraries, which allow the application writer to communicate with external systems. These libraries include:
    - The Java Database Connectivity (JDBC) API for database access
    - Java Naming and Directory Interface (JNDI) for lookup and discovery
    - RMI and CORBA for distributed application development
    - JMX for managing and monitoring applications
  + User interface libraries, which include:
    - The Abstract Window Toolkit (AWT), which provides GUI components, the means for laying out those components and the means for handling events from those components
    - The Swing libraries, which are built on AWT but provide implementations of the AWT widgetry
    - APIs for audio capture, processing, and playback
* A platform dependent implementation of Java Virtual Machine that is the means by which the byte codes of the Java libraries and third party applications are executed
* Plugins, which enable applets to be run in Web browsers
* Java Web Start, which allows Java applications to be efficiently distributed to end-users across the Internet
* Licensing and documentation.

### 4.2.6 Editions

Sun has defined and supports four editions of Java targeting different application environments and segmented many of its APIs so that they belong to one of the platforms. The platforms are:

* Java Card for smartcards.
* Java Platform, Micro Edition (Java ME)
* Java Platform, Standard Edition (Java SE)
* Java Platform, Enterprise Edition (Java EE)

The classes in the Java APIs are organized into separate groups called packages. Each package contains a set of related interfaces, classes and exceptions. Refer to the separate platforms for a description of the packages available.

The set of APIs is controlled by Sun Microsystems in cooperation with others through the Java Community Process program. Companies or individuals participating in this process can influence the design and development of the APIs. This process has been a subject of controversy.

Sun also provided an edition called PersonalJava that has been superseded by later, standards-based Java ME configuration-profile pairings.

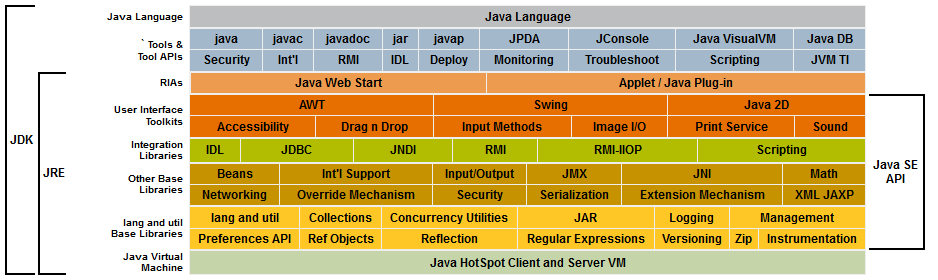


Figure 4.3Java Component Technologies

## 4.3 BlueCove Library

BlueCove is a Java library for Bluetooth (JSR-82 implementation) that currently interfaces with the Mac OS X, WIDCOMM, BlueSoleil and Microsoft Bluetooth stack found in Windows XP SP2 or Windows Vista and WIDCOMM and Microsoft Bluetooth stack on Windows Mobile. BlueCove-GPL is additional GPL licensed module to support BlueCove runtime on Linux BlueZ. BlueCove JSR-82 Emulator module is additional module for BlueCove to simulate Bluetooth stack.

BlueCove can be used in Java Standard Edition (J2SE) 1.1 or newer.

BlueCove has been tested on following JVM:

* Apache Harmony - Open Source Java SE 5.0M5 on Win XP 32 bit
* SUN Java 1.1.8 on Windows XP/X86
* SUN Java 2 Platform Standard Edition 1.4.2 on Windows XP/X86
* SUN Java 2 Platform Standard Edition 5.0 on Windows XP/X86
* SUN Java 2 Platform Standard Edition 6.0 on Windows XP/X86 and Windows Vista
* SUN Java Toolkit for CDC 1.0 on Windows XP/X86 (Need custom made dlls with \_\_cdecl exports)
* WebSphere Everyplace Micro Environment 5.7.2, CDC 1.0/Foundation 1.0/Personal Profile 1.0 on Windows XP/X86
* WebSphere Everyplace Micro Environment 6.1.1, CDC 1.0/Foundation 1.0/Personal Profile 1.0 on Windows XP and Linux/X86
* WebSphere Everyplace Micro Environment 5.7.2, CLDC 1.1, MIDP 2.0 on Windows XP/X86
* WebSphere Everyplace Micro Environment 6.1.1, CLDC 1.1, MIDP 2.0 on Linux/X86
* Mysaifu JVM - An open-source (GPL v.2 license) Java VM on Windows Mobile 2003 Second Edition.
* Java for Mac OS X 10.4 and 10.5, Release 5 on PowerPC and Intel processors
* SUN Java 2 SE 5.0 on Linux Fedora 6 i386 BlueZ 3.7, Fedora 7 i386 BlueZ 3.9, (Also tested on Ubuntu and OpenSUSE, i386 and 64-bit)
* GNU libgcj 4.1.2 on Fedora 7 i386 BlueZ 3.9
* SableVM on Debian 4.0r3 ARM on Linksys NSLU2

BlueCove provides JSR-82 Java interface for following Bluetooth Profiles:

* SDAP - Service Discovery Application Profile
* RFCOMM - Serial Cable Emulation Protocol
* L2CAP - Logical Link Control and Adaptation Protocol
* OBEX - Generic Object Exchange Profile (GOEP) profile on top of RFCOMM and TCP

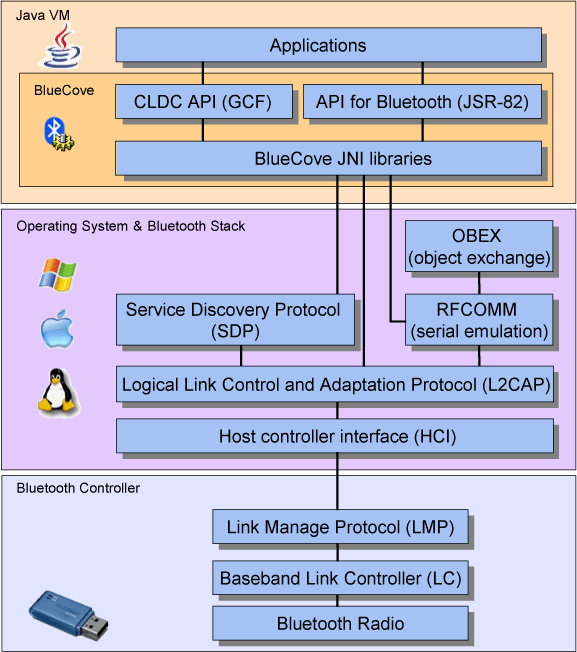


Figure 4.4 BlueCove Architecture

## 4.4 Eclipse IDE

Eclipse is an open-source community that develops open platforms and products. The community says its projects "are focused on building an open development platform consisting of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle". The Eclipse Foundation is a non-profit corporation which acts as the steward of the Eclipse community. Eclipse community states, "Eclipse means a lot of different things to different people. To some Eclipse is a Java development environment. To others, Eclipse is a flexible environment to experiment with new computer languages or extensions to existing languages." In the software world, a simple mention of "Eclipse" usually refers to the Eclipse SDK. The Eclipse SDK consists of the Eclipse Platform, Java development tools and the Plug-in Development Environment. The Eclipse Platform is a multi-language software development environment comprising an IDE and an extensible plug-in system. It is written mostly in Java. By means of various plug-ins, it can be used to develop applications in various programming languages including Ada, C, C++, COBOL, Erlang, Java, Perl, PHP, Python, R, Ruby, Scala, Clojure, Groovy and Scheme. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others. The initial codebase originated from VisualAge. The Eclipse SDK is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Released under the terms of the Eclipse Public License, Eclipse SDK is free and open source software. It was one of the first IDEs to run under GNU Classpath and it runs without issues under IcedTea.

### 4.4.1 History

Eclipse began as an IBM Canada project. Object Technology International (OTI), which had previously marketed the Smalltalk-based VisualAge family of IDE products, developed the new product as a Java-based replacement. In November 2001, a consortium was formed to further the development of Eclipse as open-source software. In January 2004, the Eclipse Foundation was created. Eclipse 3.0 (released on 21 June 2004) selected the Open Services Gateway initiative (OSGi) Service Platform specifications as the runtime architecture.

### 4.4.2 Licensing

The Eclipse Public License (EPL) is the fundamental license under which Eclipse projects are released. Some projects require dual licensing, for which the Eclipse Distribution License (EDL) is available, although use of this license must be applied for and is considered on a case-by-case basis.The Eclipse SDK was originally released under the Common Public License, but was later relicensed under the Eclipse Public License. The Free Software Foundation has said that both licenses are free software licenses, but are incompatible with the GNU General Public License (GPL). Mike Milinkovich, of the Eclipse Foundation commented that moving to the GPL would be considered when version 3 of the GPL was released.

### 4.4.3 Releases

Since 2006, the Foundation has coordinated an annual Simultaneous Release. Each release includes the Eclipse Platform as well as a number of other Eclipseprojects.So far, each Simultaneous Release has occurred on the fourth Wednesday of June.

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Codename | Date | Platform Version |
| 1 | NA | 21 June 2004 | 3.0 |
| 2 | NA | 28 June 2005 | 3.1 |
| 3 | Callisto | 30 June 2006 | 3.2 |
| 4 | Europa | 29 June 2007 | 3.3 |
| 5 | Ganymede | 25 June 2008 | 3.4 |
| 6 | Galileo | 24 June 2009 | 3.5 |
| 7 | Helios | 23 June 2010 | 3.6 |
| 8 | Indigo | 22 June 2011 | 3.7 |
| 9 | Juno | 27 June 2012 | 3.8 and 4.2 |
| 10 | Kepler | June 2013 | 4.xx |

Table 4.2 Different Releases of Eclipse IDE

### 4.4.4 Architecture

The Eclipse Platform uses plug-ins to provide all functionality within and on top of the runtime system, in contrast to some other applications, in which functionality is hard coded. The Eclipse Platform's runtime system is based on Equinox, an implementation of the OSGi core framework specification.

This plug-in mechanism is a lightweight software componentry framework. In addition to allowing the Eclipse Platform to be extended using other programming languages such as C and Python, the plug-in framework allows the Eclipse Platform to work with typesetting languages like LaTeX, networking applications such as telnet and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK, with support for other version control systems provided by third-party plug-ins.

With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every plug-in developed integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are "created equal". Eclipse provides plug-ins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plug-ins include a UML plug-in for Sequence and other UML diagrams, a plug-in for DB Explorer, and many others.

The Eclipse SDK includes the Eclipse Java development tools (JDT), offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat filespace allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards.

Eclipse implements widgets through a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also uses an intermediate graphical user interface layer called JFace, which simplifies the construction of applications based on SWT.

### 4.4.5 Rich Client Platform

Eclipse provides the Rich Client Platform (RCP) for developing general purpose applications. The following components constitute the rich client platform:

* Equinox OSGi
* Core platform
* Standard Widget Toolkit (SWT)
* JFace
* Eclipse Workbench

**Chapter 5**

# System Implementation

## 5.1 Overview of the Project

The main objective of the project is to control the PC from afar. The computer is connected to the mobile phones using Bluetooth. The mobile phone will control the PC by emulating a mouse, keyboard and simple joystick. It also provides shortcut keys to control power point presentations and any videos playing in a media player.Communication is established using the Bluetooth modules. Only paired devices can establish the connection.Once connection is successfully completed, any of the modules can be invoke to control the computer.

## 5.2 Module Description

There are 2 major modules that are implemented in this project.

### 5.2.1 PC Server Module

This is an application that runs on the PC, it will help establish communication to the phone using Bluetooth. It performs the following operations:

* It will allow incoming Bluetooth signals from the mobile
* It converts the signals into suitable actions
* It allows only one phone to control the PC

### 5.2.2 Mobile Client Module

This app is installed in the Android powered mobile phone. It connects to a paired PC via Bluetooth. It performs following operations:

* It will request the user to turn on Bluetooth
* It will connect to a paired PC to control it
* After establishing connection, any of the sub modules can be used.
* Exiting the application will close the connection

There are four sub modules within the Mobile Client Module. They are:

#### 5.2.2.1 Mouse Module

This module is used to emulate a mouse and help control the mouse in the computer. It performs the following operations:

* It gets the user’s input and sends it to the PC
* It performs mouse movement, left click, right click and left double click.
* Mouse sensitivity can be set in the settings menu

#### 5.2.2.2 Keyboard Module

This module is used to emulate a keyboard and help control the keyboard in the computer. It performs the following operations:

* A keyboard will pop once the keyboard tab is selected.
* All alphanumeric keys can be emulated
* The Shift, Alt and Ctrl button emulate the buttons on the keyboard
* The volume keys will change the focus of applications that are open
* When a key on the keyboard is pressed, it will be emulated on the PC

#### 5.2.2.3 Shortcuts Module

This module is used to provide shortcut keys for some popular applications. It performs the following operations:

* It provides general shortcuts for PowerPoint Presentation.
* They are play, next, previous, quit and black screen
* It also provides general shortcuts for some popular Media Players
* The shortcuts are playback controls and volume controls.
* Media Players supported are Zoom Player, VLC Media Player and Windows Media Player

#### 5.2.2.4 Joystick Module

This module is used to emulate a simple joystick to play simple games. It performs the following operations:

* It has 4 directional buttons and 2 playable buttons.
* The Search button emulate the Start button
* Once connection has established, the slider should at maximum to enable the joystick
* The buttons keys can be changed in the Settings menu
* It gets the user input and sends it to the PC Server app

**Chapter 6**

# System Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, coding. The engineer creates a serious of test cases that are intended to “demolish” the software that has been built. In fact, testing is the one step in this software engineering process that could be viewed as destructive rather than constructive.

In this project, the iterative and incremental development life cycle has been used. Therefore, after every module was implemented, the module is tested.

First, unit testing is performed on the module created. Here, the functionality of the module is tested. The module is made sure that it works for its intended purpose.

Once unit testing is done, other parts of the application are tested for its traits after implementing the module. Thus integration testing is performed after each module is inserted.

Test cases are created at these major points:

* Creating the interface
* Creating the server
* Implementing the Bluetooth connnection
* Adding the mouse module
* Adding the shortcut module
* Adding the keyboard module
* Adding the joystick module

After all the necessary modules are added, a system test is performed on the project. The accomplishment of all objectives is checked. Quality assurance is tested until a desired target is achieved. After that the project is then deployable and checked with other mobile phones and other PCs.

**Chapter 7**

# Conclusion

The “**PC CONTROL USING ANDROID MOBILE PHONE THROUGH BLUETOOTH**” project will control the Personal Computer and it performs the emulation of keyboard, mouse, and simple joystick operations. It also provides shortcuts for PowerPoint Presentation and some Media Players. Those operations are control from a distance a far. The goals achieved are:

* Control the Personal Computer through Bluetooth
* Very simple to port to android mobile phones.
* Implemented for touch screen phones.
* Usage of Battery is low
* No need to waste data charges

## 7.1 Future Works

The possible extensions that can extend this project are

1. A better user interface for the joystick module
2. Provide shortcuts for more media players
3. Run the particular application when a shortcut is pressed
4. Provide a call filter for the app
5. Switch off Bluetooth when the app exits

# Appendix 1

# Sample Coding

**Communitcation.java**

import java.io.\*;

importjava.util.\*;

importandroid.bluetooth.\*;

importandroid.os.\*;

public class Communication {

public Communication(Handler h) {

hn = h;

btadapter = BluetoothAdapter.getDefaultAdapter();

state = UNCONNECT;

}

public synchronized void CreateConn(String DeviceAddress) {

if (state == CONNECTING) {

if (ct != null) {

ct.cancel();

ct = null;

}

}

if (cmt != null) {

cmt.cancel();

cmt = null;

}

hn.sendEmptyMessage(0);

ct = new ConnectThread(DeviceAddress);

ct.start();

state = CONNECTING;

}

public synchronized void stop() {

if (ct != null) {

ct.cancel();

ct = null;

}

if (cmt != null) {

cmt.cancel();

cmt = null;

}

state = UNCONNECT;

}

public void writeData(byte[] data) {

if (state != HOLD) {

CommThread r;

synchronized (this) {

if (state != CONNECTED)

return;

r = cmt;

}

r.Write(data);

}

}

public void connectionLost() {

state = UNCONNECT;

Message msg = hn.obtainMessage(4);

Bundle bundle = new Bundle();

bundle.putString("warning", "Device connection was lost");

msg.setData(bundle);

hn.sendMessage(msg);

}

}

**MouseSim.java**

importandroid.view.GestureDetector.\*;

import android.view.\*

importandroid.view.View.\*;

importandroid.widget.Button;

public class MouseSim implements OnGestureListener, OnDoubleTapListener,

OnKeyListener {

publicMouseSim(Communication c) {

comm = c;

}

publicbooleanonDoubleTap(MotionEvent arg0) {

byte[] b = new byte[4];

b[0] = ControllerClient.MOUSE\_SIM;

b[1] = DOUBLETAP;

b[2] = RESERVED;

b[3] = RESERVED;

comm.writeData(b);

return true;

}

publicbooleanonSingleTapConfirmed(MotionEvent arg0) {

byte[] b = new byte[4];

b[0] = ControllerClient.MOUSE\_SIM;

b[1] = SINGLETAP;

b[2] = RESERVED;

b[3] = RESERVED;

comm.writeData(b);

return false;

}

publicbooleanonFling(MotionEvent arg0, MotionEvent arg1, float arg2,

float arg3) {

return false;

}

publicbooleanonScroll(MotionEvent arg0, MotionEvent arg1, float arg2,

float arg3) {

byte[] b = new byte[4];

b[0] = ControllerClient.MOUSE\_SIM;

b[1] = SCROLL;

b[2] = (byte) arg2;

b[3] = (byte) arg3;

comm.writeData(b);

return true;

}

publicbooleanonSingleTapUp(MotionEvent arg0) {

return false;

}

publicbooleanonKey(View v, intkeyCode, KeyEvent event) {

return false;

}

}

**Executor.java**

importjava.awt.\*;

importjava.awt.event.KeyEvent;

public class Executor {

public Executor() {

try {

r=new Robot();

} catch (AWTException e) {

e.printStackTrace();

}

}

public void MouseMove(double x, double y, double std, byte[] data) {

for (inti = 0; i<Math.abs(std); i++) {

x = x - Translator.MOUSE\_SET \* ((double) data[2] / std);

y = y - Translator.MOUSE\_SET \* ((double) data[3] / std);

r.mouseMove((int) x, (int) y);

if (i % 3 == 0) {

try {

Thread.sleep(1);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public void KeyClickSim(int key) {

KeyPress(key);

KeyRelease(key);

if (Translator.shiftFlag) {

Translator.shiftFlag = false;

KeyRelease(KeyEvent.VK\_SHIFT);

}

if (Translator.altFlag) {

Translator.altFlag = false;

KeyRelease(KeyEvent.VK\_ALT);

}

if (Translator.ctrlFlag) {

Translator.ctrlFlag = false;

KeyRelease(KeyEvent.VK\_CONTROL);

}

}

}

**Tray.java**

importjava.awt.\*;

importjava.awt.event.\*;

import java.net.URL;

importjavax.swing.\*;

public class Tray {

public Tray() {

tray = SystemTray.getSystemTray();

URL iurl = BluetoothServer.class.getResource("icon.jpg");

ImageIconi = new ImageIcon(iurl);

PopupMenu menu = new PopupMenu();

menu.add(about);

menu.add(exit);

icon=new TrayIcon(i.getImage(),"running",menu);

try {

tray.add(icon);

} catch (AWTException e) {

e.printStackTrace();

}

AddListener();

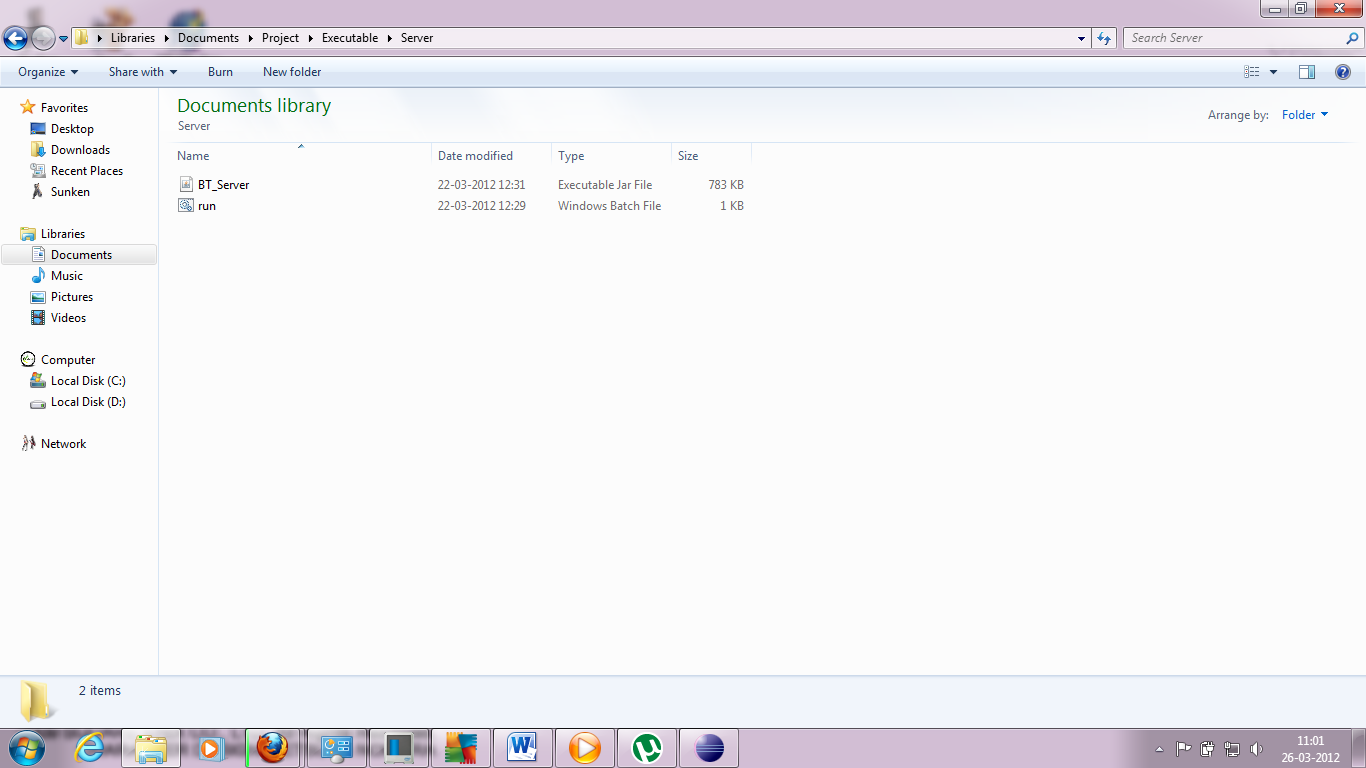
}

}

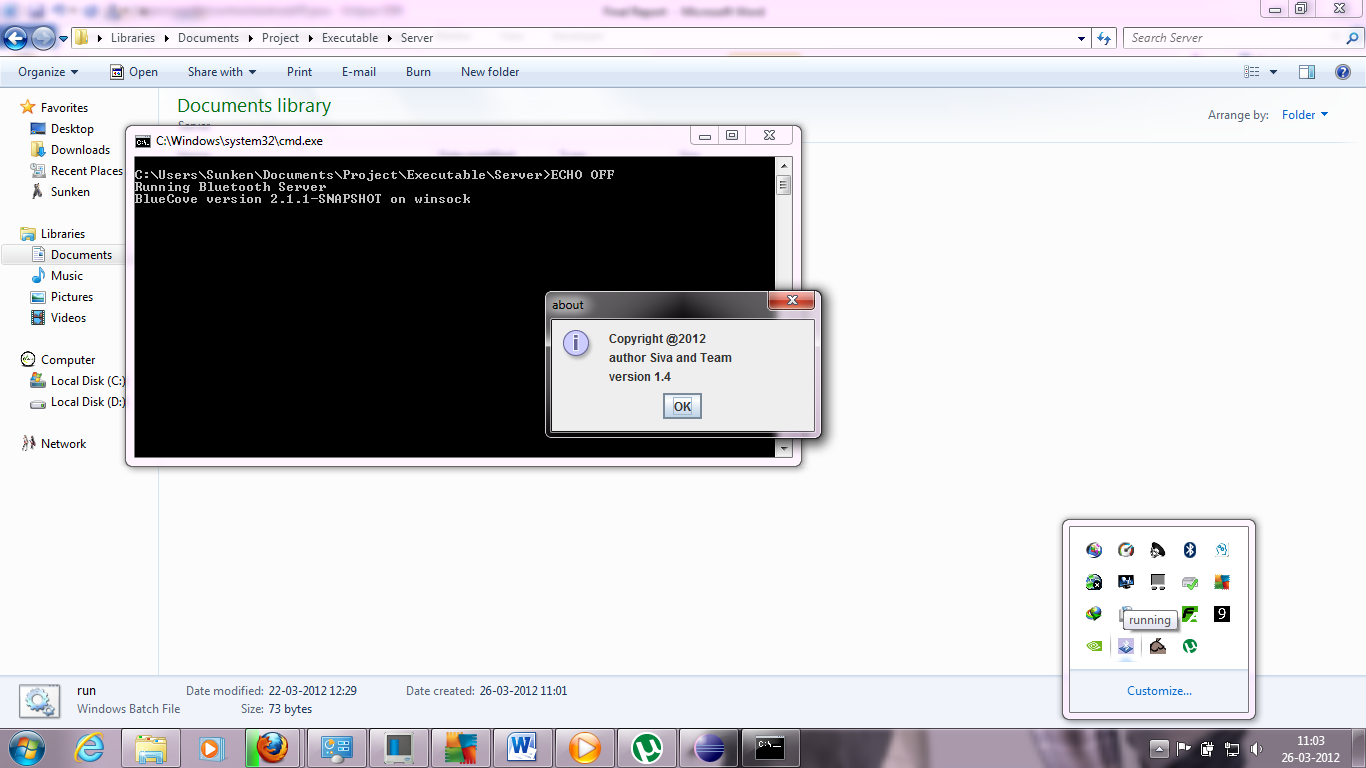
# Appendix 2

# Screenshots

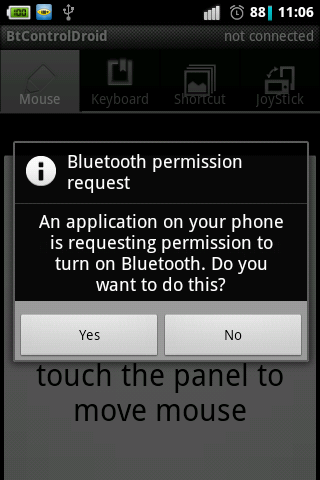
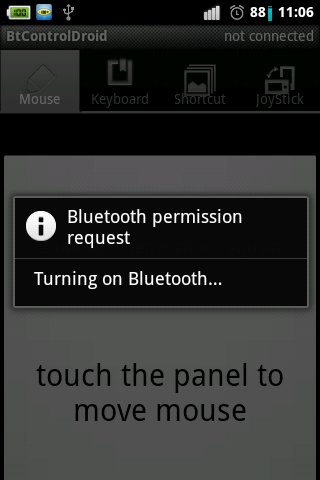
## PC Server Screenshot

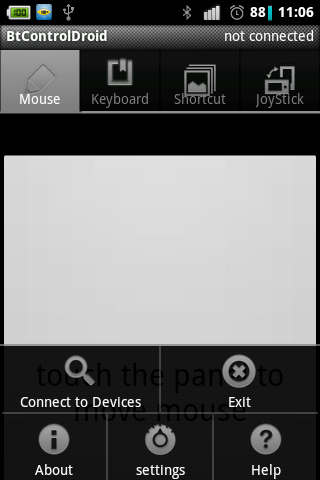
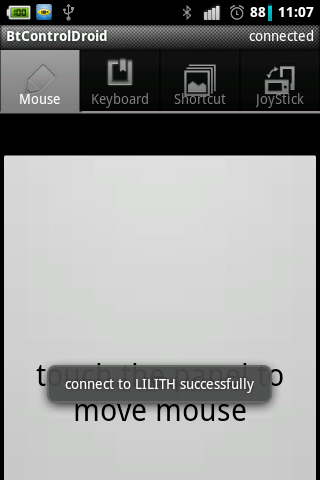


## PC Server Screenshot

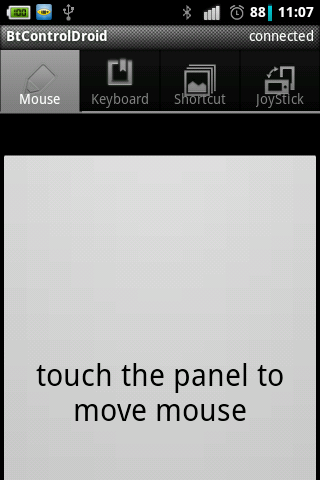


## Mobile Client Screenshot

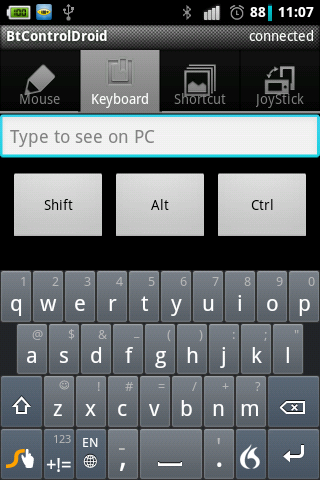
 

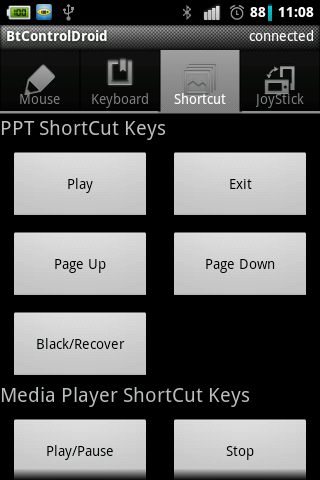
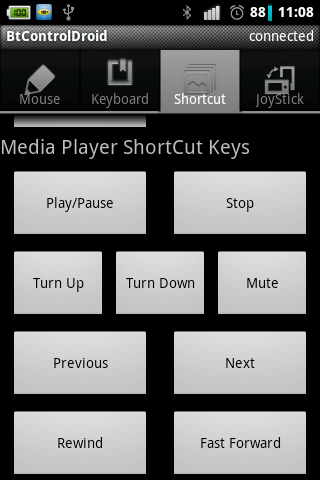
## Mouse Module Screenshot



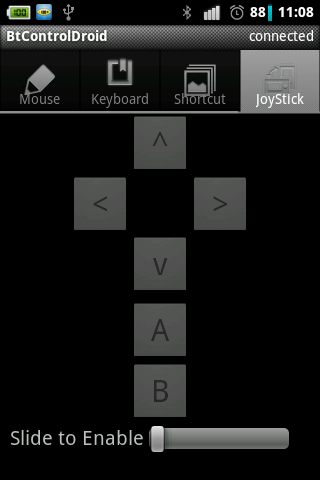
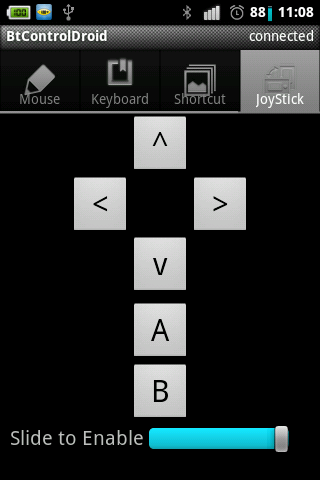
## Keyboard Module Screenshot



## Shortcut Module Screenshot

## Joystick Module Screenshot

# References

1. Barrera, David, and P.C. Van Oorschot (2010) "Secure Software Installation on Smartphones." *Security & Privacy, IEEE* 9.3 pp.42-48.
2. Chen, Eric Y., and Mistutaka, Itoh (2010) "Virtual Smartphone over IP." *World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a” ,*pp.1-6.
3. Lee, Wei-Meng (2011) *Beginning Android Application Development*. Indianapolis, IN: Wiley Pub., 2011.
4. Meier, Reto (2009) *Professional Android Application Development*. Indianapolis, IN: Wiley, 2009.
5. "Android (operating System)." *Wikipedia, the Free Encyclopedia*. 25 Dec. 2011. Web. 26 Dec. 2011.

<http://en.wikipedia.org/wiki/Android\_(operating\_system)>.

1. "The Developer's Guide." *Android Developers*. Google Inc., 21 Dec. 2011. Web. 26 Dec. 2011.

<http://developer.android.com/guide/index.html>.

1. BlueCove Team. "BlueCove - BlueCove JSR-82 Project." *BlueCove*. 25 Dec. 2008. Web. 26 Mar. 2012. <http://www.bluecove.org>.
2. "Eclipse (Software)." *Wikipedia*. Wikimedia Foundation, 25 Mar. 2012. Web. 26 Mar. 2012. <http://en.wikipedia.org/wiki/Eclipse\_(software)>.
3. "Java (programming Language)." *Wikipedia*. Wikimedia Foundation, 25 Mar. 2012. Web. 26 Mar. 2012.

<http://en.wikipedia.org/wiki/Java\_(programming\_language)>.