Atal Bihari Vajpayee - Indian Institute of Information Technology & Management
Gwalior

Project Report On

IOT based application on remotely accessing PC System using Smartphone

Submitted To:

Dr. Vinay Singh

Submitted By:

Akanksha Goel(2014-IPG-005)

Devansh Verma(2014-IPG-036)

Rahul Sant(2014-IPG-064)

Jayant Mundhra(2014-IPG-119)

Acknowledgment

We whole-heartedly thank Dr. Vinay Singh Sir for purveying the much needed guidance and mentorship to us budding minds to be able to create this project by applying various concepts of IoT taught to us.

We have learnt a lot during the development of this project and we thank Dr. Vinay Singh Sir for the same.

Regards,

Akanksha Goel

Devansh Verma

Rahul Sant

Jayant Mundhra

Table of Contents

Objective:

The aim of the project is to create an IoT empowered system where several operations like power, video playback and mouse can be remotely controlled by a smart phone device.

Similar Works:

There are a large number of implementations to the solution applied to Android software stack. It has an open protocol and it is widely deployed in the open source community. This solution adapts very well to provide part of the functionality of the architecture, and it will be studied further.

Lingyan Bi et al. proposed a novel method to Design a Android based Remote Control System e with JNI Interface for providing convenience for the user. Michael Spreitzenbarth et al. proposed analysis based Smartphone Mobile Malware for forensic Analyses. Xinfang Lee, et al. presented a novel Android based Forensic System.

Enck, W et al. proposed a secure Android Remote controlling mechanism for performing secure transaction form the Remote location. T. Richardson et al. proposed a novel method of Internet based Android application to demonstrate working of Internet Computing.

The growing popularity and spread of smart phones has changed the design of computer systems as they were known in recent years. Technological developments have enabled the creation of mobile devices with technical features previously only conceived in PC architectures or similar devices.

With this evolution comes the need to integrate these devices with others so they can take actions and monitor interaction on mobile devices .Other aspect to be considered is the remote visualization mechanisms that are useful for achieve a remote display of the devices. The most popular system designed to perform remote control of devices is Virtual Networking Computing.

Applications in Similar Domain

Airdroid (Pc to mobile connection) - AirDroid is a free and fast Android application that helps you manage your Android phone/tablet from a web browser, all over the air.

ScreenShare(Mobile to Pc connection) - ScreenShare enables wireless sharing of videos, music, photos, web pages and many types of documents from your phone to your PC.

Project need / Motivation:

With the development of technology and the continuous improvement of people's living standard,people are in pursuit of automated, intelligent and convenient home control systems. At present, the PC is used as the remote control terminal for most home control systems; however, there are some problems in the PC monitor terminal, such as its great bulk, inconvenience to carry, high cost, limited monitoring range and so on. Therefore, it"s a good choice to design a terminal based on "**phone**". With the popularity of smart phones, particularly, the phone based on Android system is rapidly developed. It means that the remote control based on Android phone will become a mainstream way. The purpose of the document is to access the PC with the smart phone. A person sitting far away from the PC can control it without moving from his/her place.

We got the motivation for the project from the real time experience of the people using the PC. Through this application we have not only made watching the videos easy but also can a person use mouse pad while he/she is away from the PC or shutdown it down at once.

Distinction of Work:

- You can find many applications related to Pc to Pc connection or an application to run a mobile from Pc but there are limited numbers of applications that help in connection to run Pc applications from mobile. This app is one of latter one.
- Our application doesn’t require net connectivity other than similar application Airdroid and MobiZen requires net connectivity for Lan based Communication. This helps in Remote connectivity that is we just require Pc and mobile for connection.

Requirement specification:

Following are the major requirements that are used in the project:

- A smartphone(with OS above version 4)
- A Computer
- Eclipse/Server
- Jdk (Java Development Kit)
- Android Package

The user needs to install an android application. After installation the code is run in the computer itself. For a successful connection the user is required to know the IP address of the computer as well as the port for activation of the services.

Once the devices are successfully connected, various features of the app can be used.

Input set and source:

Input Source:

The project derives various kinds of variegated input commands from the touch screen interface of the smart phone device. The screen would be embraced by various buttons on it which would be dedicated to perform certain functions. And, upon touching any of the buttons the dedicated command and operation will be executed.

Input Set:

Remote power control operations.	Shut Down
	Sleep
	Restart
Remote mouse control operations.	Right Click
	Left Click
	Scroll
Remote VLC control operations.	Play
	Pause
	Next
	Previous
	Volume Up
	Volume Down

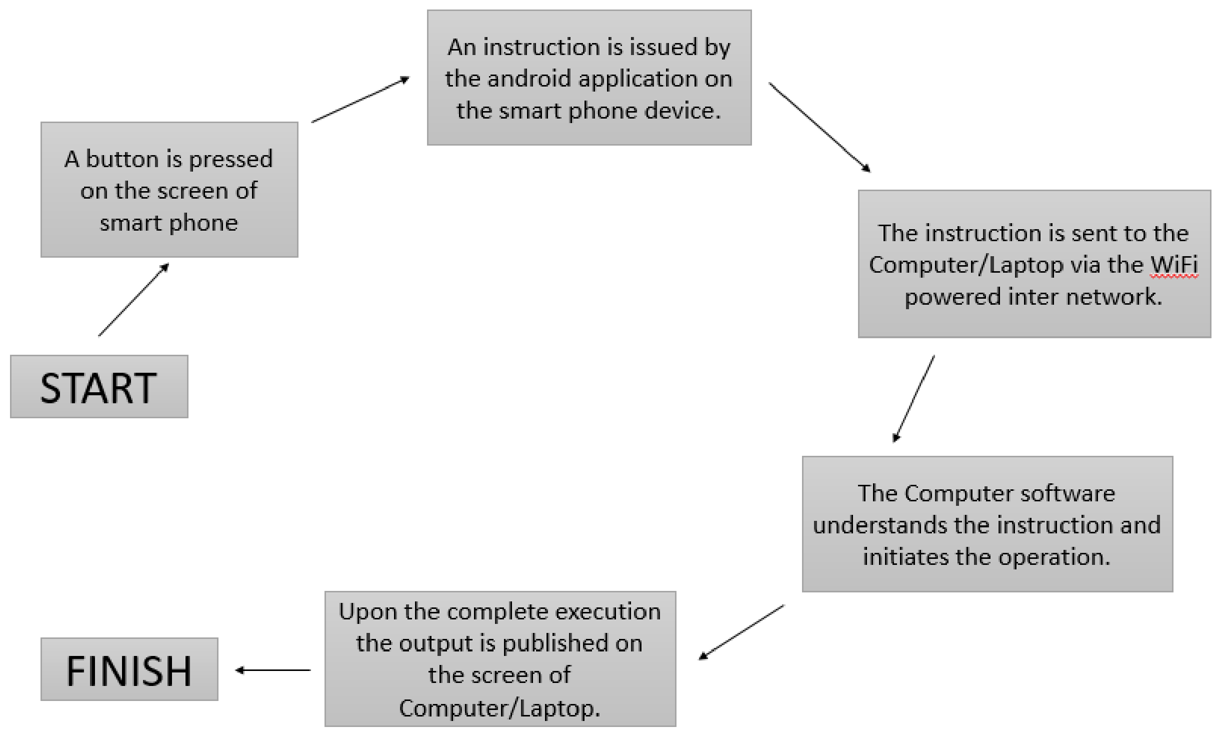
Targeted output sets:

Remote power control operations.	Shut Down	The computer/laptop device shuts down.
	Sleep	The computer/laptop device goes to sleep.
	Restart	The computer/laptop restarts.
Remote mouse control operations.	Right Click	The right click command will be executed on the computer/laptop screen.
	Left Click	The left click command will be executed on the computer/laptop screen.
	Scroll	The user will be able to scroll up and down using his fingers.
Remote VLC control operations.	Play	The play command would play the video/music being run on VLC media player.
	Pause	The pause command would play the video/music being run on VLC media player.

	Next	The next command would play the next video/music in the playlist on VLC media player.
	Previous	The next command would play the next video/music in the playlist on VLC media player.
	Volume Up	The Volume Up command would increment the volume.
	Volume Down	The volume Down command would decrement the volume.

Data Flow Model:

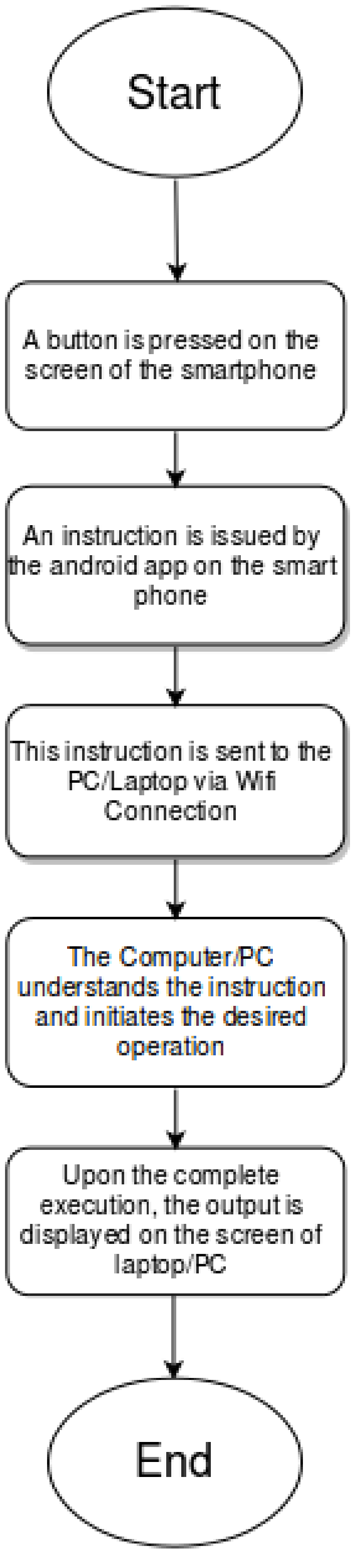
The data would flow in the form of instructions that will be activated by the user by means of various options available on the touch screen interface. Once an instruction is issued, the data will flow from the smart phone device to the computer/laptop device by the means of an inter-network established between the two hardware devices with the help of a Wi-Fi hotspot connection.


START
A button is pressed on the screen of smart phone
An instruction is issued by the android application on the smart phone device.
The instruction is sent to the Computer/Laptop via the WiFi powered inter network.
The Computer software understands the instruction and initiates the operation.
Upon the complete execution the output is published on the screen of Computer/Laptop.
FINISH

Process Flow Model:

Each process consists of various conditions that would have to be fulfilled to be executed. Thus, whenever a process will be executed, the data associated with it will be transferred to the computer/laptop device where it will lead to the various sub-routines/sub-processes which would finally completely execute the desired process.

Thus, a process will be executed by the flow of precedence from one sub-process to another.


Start
A button is pressed on the screen of the smartphone
An instruction is issued by the android app on the smart phone
This instruction is sent to the PC/Laptop via Wifi Connection
The Computer/PC understands the instruction and initiates the desired operation
Upon the complete execution, the output is displayed on the screen of laptop/PC
End

Coding/Implementation part

Creating Client App: As this example is meant for only for explaining the concept, we'll create a very simple one page android app. It has only one activity which has 8 controls and two Textboxes which takes Host computer's IP Address and server port as input. One TextView which will act as our mouse pad and 7 buttons (Next, Play/Pause, Previous,Lock,ShutDown,Sleep,Restart).

Create a new android project by the name PCRemote, with a blank activity (activity_main). Create the required textview and buttons either using the design view or text view. You can copy paste the following code to create the required activity layout.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context="com.example.dell15.unifiedremote.MainActivity1">

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Next"
    android:id="@+id/button2"
    android:layout_above="@+id/editText2"
    android:layout_toRightOf="@+id/button3"
    android:layout_toEndOf="@+id/button3"
    android:layout_margin="0dp"
    android:layout_marginRight="2dp"
    android:paddingLeft="0dp"/>

  <EditText
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:inputType="number"
    android:ems="10"
    android:id="@+id/editText"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignRight="@+id/editText2"
    android:layout_alignEnd="@+id/editText2" />

<EditText
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editText2"
    android:width="400dp"
    android:layout_above="@+id/editText"
    android:layout_alignRight="@+id/button"
    android:layout_alignEnd="@+id/button" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Previous"
    android:id="@+id/button3"
    android:layout_above="@+id/editText2"

    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginLeft="0dp"
    android:layout_margin="0dp"
    android:paddingLeft="0dp"
    android:background="#8f968c" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Play/Pause"
    android:id="@+id/button"
    android:layout_alignTop="@+id/button2"
    android:layout_toRightOf="@+id/button2"
    android:layout_toEndOf="@+id/button2" />

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:id="@+id/textView2"
    android:layout_alignParentTop="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_above="@+id/button2"
    android:background="#a91d1d" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Connect"
    android:id="@+id/button4"
    android:layout_below="@+id/button2"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ShutDown"
    android:id="@+id/button9"
    android:layout_alignTop="@+id/editText"
    android:layout_alignRight="@+id/button4"
    android:layout_alignEnd="@+id/button4" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Restart"
    android:id="@+id/button11"
    android:layout_below="@+id/button2"
    android:layout_alignRight="@+id/editText"
    android:layout_alignEnd="@+id/editText"
    android:layout_marginRight="25dp" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sleep"
    android:id="@+id/button12"
    android:layout_alignParentBottom="true"
    android:layout_toLeftOf="@+id/button4"
    android:layout_toStartOf="@+id/button4"
    android:layout_marginRight="25dp" />

<Button
```

```
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Lock"
    android:id="@+id/button13"
    android:layout_above="@+id/editText2"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />

</RelativeLayout>
```

Main App

To send data to server, we need to create a stream socket and connect it to pre-determined port number and server IP address.

```
InetAddress serverAddr = InetAddress.getByName("SERVER_IP");
Socket socket = new Socket(serverAddr, PORT);
```

Once, the socket is connected, we need to create an output stream on that socket to send data to the server. Anything that we write on that output stream will be sent to the server.

```
PrintWriter outStream = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())), true);
outStream.println("Hi") //This will send "Hi" to server
```

When the user taps on play, next or previous button, we send "play", "next" or "previous" text respectively to the server using outStream.println() and when the user drags on the mousePad text view, we send the movement in x and y direction in "x,y" format. To implement this complete functionality, paste the following code in your MainActivity.java file

Code:

```
package com.example.dell15.unifiedremote;

import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.AsyncTask;
import android.os.IBinder;
import android.provider.SyncStateContract;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.InetSocketAddress;
import java.net.Proxy;
import java.net.Socket;
import java.net.UnknownHostException;

import static android.widget.Toast.LENGTH_SHORT;
interface AsyncTaskCompleteListener<T> {
  public void onTaskComplete(PrintWriter result);
}

public class MainActivity1 extends AppCompatActivity implements View.OnClickListener,
View.OnTouchListener,AsyncTaskCompleteListener<PrintWriter> {

  Button b1, b2, b3, b4,b5, b6 ,b7, b8;
  private boolean isConnected = false;
  //private boolean mouseMoved=false;
  private Socket socket;
  private PrintWriter out;
  EditText t1, t2;
  TextView q1;
  private static int port;
  private static String server_name;
  private boolean mouseMoved=false;
  private float initX =0;
  private float initY =0;
  private float disX =0;
  private float disY =0;
  private long lastTouchDown;
  private static int CLICK_ACTION_THRESHHOLD = 200;
  public static boolean n1=false;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main1);
    b1 = (Button) findViewById(R.id.button);
    b2 = (Button) findViewById(R.id.button2);
    b3 = (Button) findViewById(R.id.button3);
    t1 = (EditText) findViewById(R.id.editText);
    t2 = (EditText) findViewById(R.id.editText2);
    b4 = (Button) findViewById(R.id.button4);
    q1 = (TextView) findViewById(R.id.textView2);
    b5 = (Button) findViewById(R.id.button9);
    b6 = (Button) findViewById(R.id.button11);
    b7 = (Button) findViewById(R.id.button12);
    b8 = (Button) findViewById(R.id.button13);
    b1.setOnClickListener(this);
    b2.setOnClickListener(this);
    b3.setOnClickListener(this);
    b4.setOnClickListener(this);
    q1.setOnTouchListener(this);
    b5.setOnClickListener(this);
    b6.setOnClickListener(this);
    b7.setOnClickListener(this);
    b8.setOnClickListener(this);

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu1, menu);
    return true;

}


@Override
public void onClick(View view) {
    switch (view.getId()) {
        case R.id.button:
            if (isConnected && out != null)
                out.println("Play");
            break;

        case R.id.button2:
            if (isConnected && out != null)
                out.println("Next");
            break;
        case R.id.button3:
            if (isConnected && out != null)
                out.println("Previous");
            break;

        case R.id.button4:
            if(!isConnected)
            {
                server_name = t2.getText().toString();
                port = Integer.parseInt(t1.getText().toString());
                launchTask(server_name,port,isConnected);

            }
            break;

        case R.id.button9:
            if (isConnected && out != null)
                out.println("ShutDown");
            break;

        case R.id.button11:
            if (isConnected && out != null)
                out.println("Restart");
            break;

        case R.id.button12:
            if (isConnected && out != null)
                out.println("Sleep");
            break;


        case R.id.button13:
            if (isConnected && out != null)
                out.println("Lock");
            break;

        default:
            break;

    }
}

@Override
public void onDestroy() {
    super.onDestroy();
    if (isConnected && out != null) {
        try {
            out.println("exit"); //tell server to exit
            socket.close(); //close socket
        } catch (IOException e) {
            Log.e("remotedroid", "Error in closing socket", e);
        }
    }
}

@Override
public boolean onTouch(View view, MotionEvent motionEvent)
{
    if(isConnected && out!=null)
    {
        switch(motionEvent.getAction())
        {
            case MotionEvent.ACTION_DOWN:
                initX = motionEvent.getX();
                initY = motionEvent.getY();
                lastTouchDown = System.currentTimeMillis();
                mouseMoved = false;
                break;

           case MotionEvent.ACTION_MOVE:
                disX = motionEvent.getX() - initX;
                disY = motionEvent.getY() - initY;
                initX = motionEvent.getX();
                initY = motionEvent.getY();

                if(disX!=0 && disY!=0)
                out.println(disX+","+disY);

                mouseMoved = true;
                break;

            case MotionEvent.ACTION_UP:
                if (System.currentTimeMillis() - lastTouchDown < CLICK_ACTION_THRESHHOLD)
                {
                    out.println("left_click");
                }




        }
    }
    return true;
}

@Override
public void onTaskComplete(PrintWriter out)
{System.out.println("Hello world !");
```

```
        this.out = out;

        isConnected = true;
    }

    public void launchTask(String server_name,int port,boolean isConnected) {
        MySyncTask a = new MySyncTask(server_name,port,getBaseContext(),this);
        a.execute();

    }


    public class ConnectPhoneTask extends AsyncTask<String, Void, Boolean> {

        @Override
        protected void onPreExecute() {
            server_name = t2.getText().toString();
            port = Integer.parseInt(t1.getText().toString());

        }

        @Override
        protected Boolean doInBackground(String... params) {
            boolean result = true;

            try {

                InetSocketAddress inet = new InetSocketAddress(server_name,port);
                socket = new Socket();
                socket.bind(null);
                socket.connect(inet,0);
                isConnected = true;
                out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket
                        .getOutputStream())), true);
                //onPostExecute1(true);

            } catch (Exception e) {
                System.out.println("Socket is not created");
                Log.e("remotedroid", "Error while connecting", e);
                result = false;
            }
            //onPostExecute1(result);
            return result;
        }
        public void onPostExecute1(Boolean result) {
            isConnected = result;
            Toast.makeText(getApplicationContext(), isConnected ? "You are connected" : "Error while
connecting...", LENGTH_SHORT).show();
            if (isConnected) {
```

```
            try {
                out = new PrintWriter(new BufferedWriter(new OutputStreamWriter(socket
                        .getOutputStream())), true);
            } catch (IOException e) {
                Log.e("remotedroid", "Error while creating OutWriter", e);
                Toast.makeText(getApplicationContext(), "Error while connecting",
Toast.LENGTH_LONG).show();
            }
        }
    }
  }
}
```

We can ask for server IP from the user in an EditText and then connect to that IP on a pre-determined port number or the port no. Is itself given by the server, but for this example, we have hard-coded both these values and you should replace them based on your requirement.

Writing the Server

The client just connects via an open socket and send some text over to the server. Server is where all the magic of remote control happens. To simulate keyboard and mouse events on the server, we use java.awt.Robot class. This class was built mainly for the purpose of test automation, but can be used for various other purposes.

First thing we need to do on server is to open a server socket and then wait for a client to connect to it. Let’s create a server socket on a port

```
ServerSocket serverSocket = new ServerSocket(SERVER_PORT);
Socket clientSocket = server.accept();
```

Once the client is connected and is sending data, we’ll read the input line by line in a BufferedReader and then act on the basis of each input.

```
BufferedReader in = new BufferedReader(new InputStreamReader(client.getInputStream()));
String line = in.readLine();
```

To simulate keyboard events, we’ll the keyPress and keyRelease methods of Robot class. So for example, when the user wants to move to next song in the play list, he/she taps on “Next” button in the app which sends the string “next” to the server and upon receiving this string, the server simulates the press and release of key “n” on keyboard. We are using following VLC shortcuts:-

Next – n

Previous – p

Play/Pause – space

These are default shortcuts, you need to modify code if you have changed your VLC shortcuts. And for the power options, we have used java.lang.Runtime class which takes cmd array as input and execute it. We can do a lot more with it such as opening some file or application directly from our app.

Both keyPress and keyRelease methods take an integer as input. These integers can be obtained fromKeyEvent class. KeyEvent has static fields corresponding to each keyboard key and event. This is how we simulate press and release of “n”

```
Robot robot = new Robot();
robot.keyPress(KeyEvent.VK_N);
robot.keyRelease(KeyEvent.VK_N);
```

To simulate a mouse event, we need to use the mouse related methods in Robot class. To move the mouse pointer, use Robot.mouseMove(x,y) and to simulate mouse click, use Robot.mousePress() and Robot.mouseRelease().

The complete server code can be written as below:-

```
package server1;

import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import java.awt.MouseInfo;
import java.awt.Point;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.InetSocketAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.awt.*;


class connect_server
{


    private static ServerSocket server = null;
    private static Socket client = null;
    private static BufferedReader in = null;
    private static String line;
    private static boolean isConnected=true;
    private static Robot robot;
    private static int SERVER_PORT=0 ;


    public static void main(String [] args)
    {



        for(int i=1;i<1023;i++)
        {
            try
            {
            robot = new Robot();
            server = new ServerSocket(i);
            if(server!=null)
            {
            SERVER_PORT=i;
            System.out.println("Server is created at port "+server.getLocalPort());

            break;


            }




        }
        catch (IOException e) {
            server = null;
            //System.out.println("Error in opening Socket");
            //System.exit(-1);
        }
```

```
catch (AWTException e) {
    System.out.println("Error in creating robot instance");
    System.exit(-1);
}

}
try{
    server.setSoTimeout(100000);
    System.out.println("Waiting for connection....");
client = server.accept();
in = new BufferedReader(new InputStreamReader(client.getInputStream()) );


while(isConnected)
{
    System.out.println("Connected to the client");
    try {
        line  = in.readLine();
        System.out.println(line);
        if(line.equalsIgnoreCase("Next"))
        {
            //Simulate press and release of key 'n'
            System.out.println("N is pressed");
            robot.keyPress(KeyEvent.VK_N);
            robot.keyRelease(KeyEvent.VK_N);
        }
        //if user clicks on previous
        else if(line.equalsIgnoreCase("Previous")){
            //Simulate press and release of key 'p'
            System.out.println("Prev is pressed");
            robot.keyPress(KeyEvent.VK_P);
            robot.keyRelease(KeyEvent.VK_P);
        }
        //if user clicks on play/pause
        else if(line.equalsIgnoreCase("Play")){
            //Simulate press and release of spacebar
            System.out.println("Play is pressed");
            robot.keyPress(KeyEvent.VK_SPACE);
            robot.keyRelease(KeyEvent.VK_SPACE);
        }
        else if(line.contains(","))
        {
            float disx = Float.parseFloat(line.split(",")[0]);
            float disy = Float.parseFloat(line.split(",")[1]);
            Point p = MouseInfo.getPointerInfo().getLocation();
            double newX = p.getX();
            double newY  = p.getY();
```

```
                    robot.mouseMove((int)(newX+disx), (int)(newY+disy));
                }

                else if(line.contains("left_click"))
                {
                    robot.mousePress(InputEvent.BUTTON1_DOWN_MASK);
                    robot.mouseRelease(InputEvent.BUTTON1_DOWN_MASK);
                }
                else if(line.equalsIgnoreCase("ShutDown"))
                {
                    System.out.println("ShutDown event is triggered");
                   Runtime runtime = Runtime.getRuntime();
                   Process proc = runtime.exec("Shutdown.exe -s -t 00");
                   System.exit(0);
                }
                else if(line.equalsIgnoreCase("Restart"))
                {
                    System.out.println("ShutDown event is triggered");
                   Runtime runtime = Runtime.getRuntime();
                   Process proc = runtime.exec("Shutdown.exe -r -t 00");
                   System.exit(0);
                }
                else if(line.equalsIgnoreCase("Sleep"))
                {
                    System.out.println("ShutDown event is triggered");
                   Runtime runtime = Runtime.getRuntime();
                   Process proc = runtime.exec("rundll32.exe powrprof.dll,SetSuspendState 0,1,0");
                   System.exit(0);
                }
                else if(line.equalsIgnoreCase("Lock"))
                {
                    System.out.println("ShutDown event is triggered");
                   Runtime runtime = Runtime.getRuntime();
                   Process proc = runtime.exec("Rundll32.exe User32.dll,LockWorkStation");
                   System.exit(0);
                }

            }

            catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }
}
        catch(Exception e)
        {
            e.printStackTrace();
        }
```

```
}
}
```

Save the above code in RemoteDroidServer.java file and then compile, execute using the usual Java way.

```
javac RemoteServer.java
java RemoteServer
```

This will start the server and wait for a client to connect. After starting the server, run the android app and try to connect to your server.

NOTE: - make sure you have specified the correct server IP address in the app. Also, that the port numbers are same on both server and app.

Software Development Limitation:

The Limitations of the projects are as follows:

- We have not taken care of any Security constrains during the whole project. A user who knows the IP address and the free port number can easily access the resources through the application
- The application is range bound. The connection can be established till both the devices are in the "range", which may last few meters.
- The shortcuts used to play, pause, next and previous are player dependent.

Project Input interface:

The project derives its inputs from a smart phone device which would provide an interface in the form of an android application. This application features a host of sections for each dedicated operation which will all have a number of operations to cater to various processes involved.

Project Output interface:

The project will derive its outputs on a computer/laptop device's display screen where the various operations and processes will deliver their desired results. For each of the instructions input through the smart phone device application, the operations will be executed on the back ground and finally the results will be enacted upon the display screen.

Java Classes used:

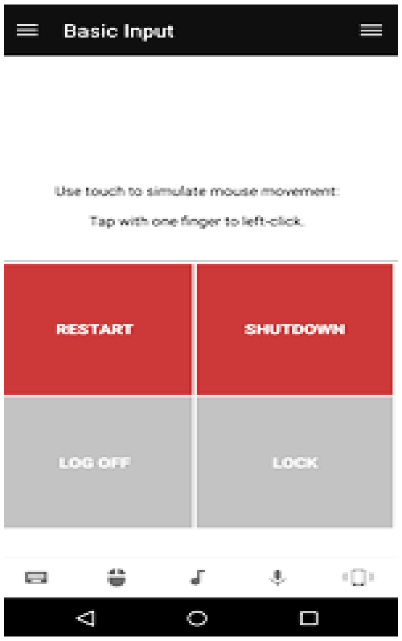
Inbuilt Classes:

1. java.io package class namely BufferedWriter, BufferedReader, IOException, OutputStream, OutputStreamWriter, InputStreamReader, InputStream, PrintWriter, PrintReader
2. java.net package classes namely InetAddress, InetSocketAddress, Proxy, Socket, UnknownHostException, InetSocketAddress, ServerSocket, Socket
3. java.awt package classes AWTException, Robot, MouseInfo, Point, event.InputEvent, event.KeyEvent

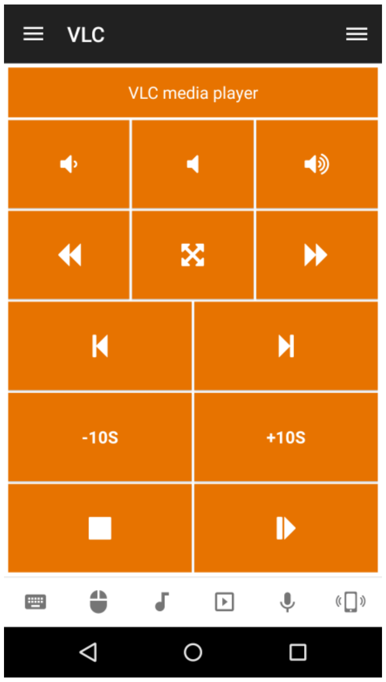
Self-Made Classes:

1. connect_server: This class is responsible for establishing the connection and transfer of data from PC to phone or vise-versa and then process the data.
2. MainActivity1: This is a class which defines all the features of the application. This class is dedicated for the User Interface

MOUSE AND POWER STIMULATIONS FROM ANDROID


Basic Input
Use touch to simulate mouse movement.
Tap with one finger to left-click.
RESTART
SHUTDOWN
LOG OFF
LOCK

VLC OUTPUT INTERFACE



Project life and maintenance:

The android application designed to provide the interface on the smart phone device will have a life dependent on the incumbent android version in power. In case the installed android version does not support the older version of our application, the application will have to be updated to be in compliance with the conditions required by the higher versions of android.

Thus, the android application will have to be regularly updated to keep it running over time.

Further, the activity and the functions of the various operations under our focus are liable to change in accordance to the external parties involved. For instance, in case the protocols for using the mouse and its programming is altered by the manufacturer, we will also have to update our software on the computer/laptop device to be compliant.

Ergo, a regular maintenance and updation will be needed to keep the application and software program in compliance with the future needs and requirements.

Conclusion:

We have created an IoT project which has two interfaces, one on the computer/laptop end, and another on the smart phone device. These software and coded interfaces create a dedicated network between the two hardware devices. Further, using the previously mentioned network we intend to control and execute certain operations of our computer/laptop device.

Our proposed work provides convenience to desktops/laptops users and save money of customers and provides best cost effective solution to their problems and key highlight is to have multitasking ability and it will also threaten the most of peripheral developer industries.

Smart phone and tablet universal remote software is usually highly customizable. As with traditional universal remotes some are programmed using the handset (phone/tablet) itself and others are programmed using a computer. Remote control features, you can finally clean up your coffee table and put your extra remotes away in a drawer somewhere.

Now your phone (or tablet) is your remote. At last your whole family (and even guests) will be able to figure out how to control all the different devices and inputs you have in the living room. A customizable remote control interface, where you decide exactly which buttons appear when you want them to.

Future Work:

Currently we have established the following operations in our project:

1. Remote power control operations.
2. Remote mouse control operations.
3. Remote keyboard control operations.
4. Remote video watching - VLC - operations.

Further we intend to expand our project by creating dedicated controls for sundry disparate applications and softwares. For example remotely watching videos over YouTube, remotely browsing the Internet and also remote sharing and transfer of data et cetera.

Also we will add Server password protection and encryption for added security and safety.

The modules will be developed keeping the user’s background in mind. The software will be made user-friendly and GUI based. Wherever required we will add the user friendly messages for easy understanding. We are planning to make a connection between multiple devices with Computer so that it can be controlled by multiple users. We are also planning screen sharing of either device to other in long run.

Thus, in our purview of expansion we intend to create a wholesome IoT empowered system where we would be able to remotely control a system from another easy to carry smart phone device.

In a nutshell, our group will be creating a Remote Access application for Android phones. This application will run on a mobile platform (Android). Which provides services for data transfer, file transfer and it has ability to view the remote location on android phones. For wireless connection Bluetooth as well as Wi-Fi are embedded to it.

Abstract:

This paper represents how your PC and Laptop can be controlled from remote place with smartphone using internet. It basically turns your smartphone into wireless mouse with touchpad and also adding a sleuth of ancillary features. This application can be performed with some wireless connection between the PC or Laptop and the smartphone with Android operating system.

By accessing the IP address of PC, we can establish a connection between them using Wi-Fi connection. This application not only turns your smartphone into wireless keyboard and mouse but it also provides various other features including voice to text conversion.

The implemented application consists of two parts, the first one is an application for Android smartphone and the second one is a server application that executes the command selected by user’s application. The outcome of this implementation is a handy, easy-to-use application.

This paper represents how your PC can be controlled from remote place with your smartphone device with the help of Internet. It means the monitor of PC will be

seen in mobile. It turns your phone into a wireless mouse with touchpad, using your own wireless network.

This application can be performed on android based mobile. It requires server application for your computer. It requires device running on the Android operating system with some sort of wireless connection between them. By getting IP address from the PC and directly browse it on mobile phone.

Thank You.