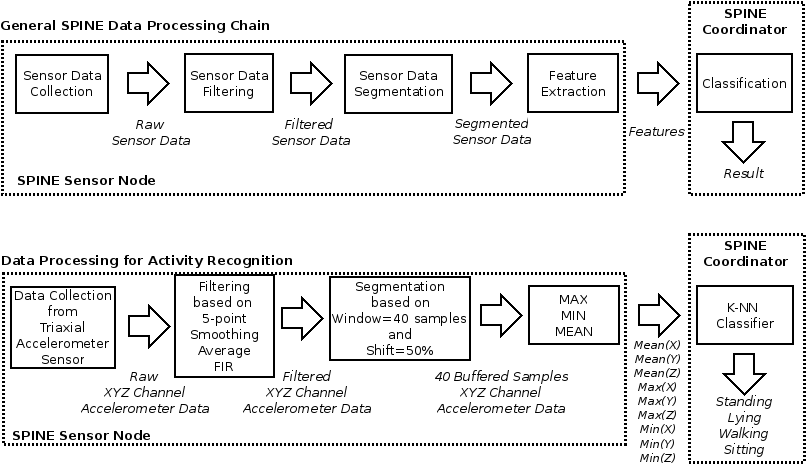
1. **Introduction**

Although touch and gesture tracking has come to the fore over the past few years, the most prevalent method of interacting with a computer is with a mouse. Track pads do the function of what a physical mouse used to, although with certain downfalls. With designing, gaming and most other tasks that are more involved than simple web browsing, most people would prefer to use a physical mouse as it allows for higher speed and accuracy. However, when using a laptop computer most people do not carry around a mouse due to the hassle of carrying around a device that is quite sizable compared to the ever shrinking computer. I have, while working in countless cafes, on countless occasions forgot to bring my mouse with me and suffered a less productive session. A few misplaced Bluetooth mouse dongles later, I thought there must be a better alternative. My proposed solution is to turn a mobile device into a mouse - much to my surprise there are no apps in the market that do this. Plenty of apps turn mobile devices into track pads but none of them translate linear device motion into cursor motion. This can be done through motion sensors, camera input analysis or a combination of the two. Further, the touch screen of a phone or the camera could be used to recognize gestures that could drive events, a benefit above the functionality of traditional mice and track pads.

The project provides the following contributions:

1. a method by which to use a mobile device as input to control computers and or other devices
2. a method by which to detect linear mobile phone movement with the level of precision and speed necessary to allow for satisfactory cursor control
3. an exploration of innovative gesture recognitionthrough both touch and video to interact with the computer.
   1. **Sensor Data Processing**



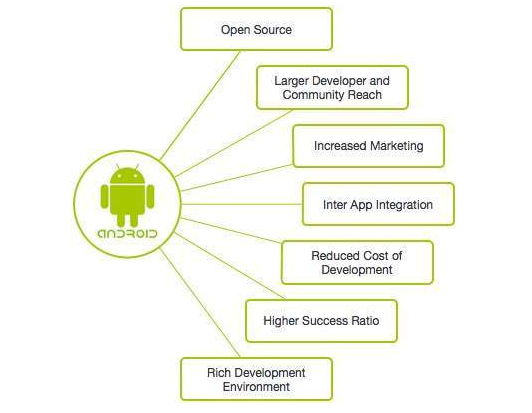
FIG\_1-Sensor Data Processing

A wireless sensor network (WSN) (sometimes called a wireless sensor and actor network (WSAN)[) are spatially distributed [autonomous](https://en.wikipedia.org/wiki/Autonomous) [sensors](https://en.wikipedia.org/wiki/Sensor) to *monitor* physical or environmental conditions,such as [temperature](https://en.wikipedia.org/wiki/Temperature), [sound](https://en.wikipedia.org/wiki/Sound), [pressure](https://en.wikipedia.org/wiki/Pressure), etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling *control* of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control, machine health monitoring, and so on.

* 1. **Android Technology**

World is contracting with the growth of mobile phone technology. As the number of users is increasing day by day, facilities are also increasing. Starting with simple regular handsets which were used just for making phone calls, mobiles have changed our lives and have become part of it. Now they are not used just for making calls but they have innumerable uses and can be used as a Camera , Music player, Tablet PC, T.V. , Web browser etc . And with the new technologies, new software and operating systems are required.

[Operating Systems](http://www.engineersgarage.com/articles/operating-systems-tutorial) have developed a lot in last 15 years. Starting from black and white phones to recent smart phones or mini computers, mobile OS has come far away. Especially for smart phones, Mobile OS has greatly evolved from Palm OS in 1996 to Windows pocket PC in 2000 then to Blackberry OS and Android.

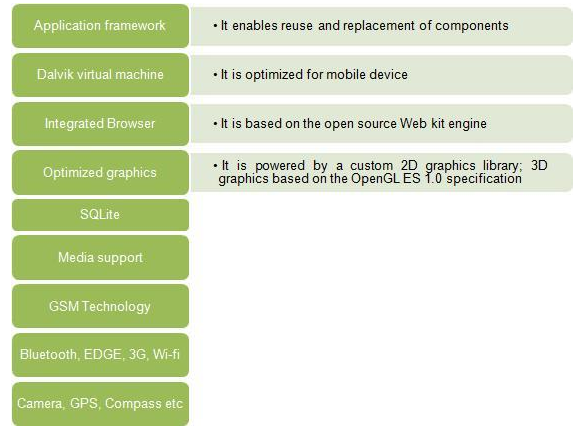


FIG\_2 Android Technology

 One of the most widely used mobile OS these days is **ANDROID**. **Android** is a software bunch comprising not only operating system but also middleware and key applications. Android Inc was founded in Palo Alto of California, U.S. by Andy Rubin, Rich miner, Nick sears and Chris White in 2003. Later Android Inc. was acquired by Google in 2005. After original release there have been number of updates in the original version of Android.

**1.2.1 Features & Specifications**

**Android** is a powerful Operating System supporting a large number of applications in [Smart Phones](http://www.engineersgarage.com/articles/smart-phones). These applications make life more comfortable and advanced for the users. Hardwares that support Android are mainly based on [ARM architecture](http://www.engineersgarage.com/articles/arm-advanced-risc-machines-processors) platform. Some of the current features and specifications of android are:



FIG\_3 Features of Android technology

Android comes with an Android market which is an online software store. It was developed by Google. It allows Android users to select, and download applications developed by third party developers and use them. There are around 2.0 lack+ games, application and widgets available on the market for users.Android applications are written in java programming language. Android is available as open source for developers to develop applications which can be further used for selling in android market. There are around 200000 applications developed for android with over 3 billion+ downloads. Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. For software development, Android provides **Android SDK** (Software development kit). Read more about [open source software](http://www.engineersgarage.com/articles/open-source-software-history-advantages).

* 1. **Objective**
* The growing popularity and spread of smartphones has changed the design of computer systems as they were known in recent years.
* Smart phone is being used by every person.
* Remote mouse using the smartphone will be very helpful.
  1. **Scope**
* Now days we can use the pc not only from limited area but using smartphone, we can access the pc within the wifi range.
* An application that control pc mouse through android device.
  1. **Why This Project?**
* Application made on android platform has more users and have less complexity.
* It’s easy to use.
* Don’t need higher skills to operate for the user.
* It’s of very low cost or we can say negligible.
* It’s reliable and user friendly.
* It’s portable

**1.6 Design Goals**

The final goal is to publish the mouse application in the App Store, and specifically target those who have an everyday need for a mouse but dislike carrying one around, or suffer from lost Bluetooth dongles. The target market would primarily include designers and gamers but would be of interest to almost anyone who prefers using a mouse to a track pad. The app provides a cheaper alternative to high tech mice, such as providing the same service with added convenience since it operates on a mobile device.

**1.7 Projects Proposed Features and Functionality**

* Wireless interface through Wi-Fi or Bluetooth between Phone and computer to swap information
* Cursor control functionality through linear motion detection, and other typical mouse controls - single-click, double-click, scrolling etc.
* Multi-touch gesture recognition similar to those detected by the Mac track pad
* One Server which is made in java to get the access of all mouse controls.

1. **System Requirement Study**
   1. **User Characteristics**

The Android mouse Application is designed for user which uses android phone.

The users who can deal with this software are:-

* User who has laptop to control the mouse.
* Android phone user

**2.2 Hardware and Software Requirements**

Hardware Requirement:-

* Android phone
* Laptop
* 512 mb ram
* 1 mb free internal storage in phone

Software Requirement:-

* Android Operating System
* Wi-Fi driver

**2.3 Tools and Technology**

* **Eclipse IDE:**

In this project I will make use of Eclipse, an IDE written primarily in Java.

In eclipse there is a provision for installing a custom plug-in called Android Development Tool?

ADT which supports creating, running, and debugging Android applications.

* **Android SDK:**

It is a software development kit from Google which includes development tools, an

Emulator and the libraries needed for building an Android application

## 

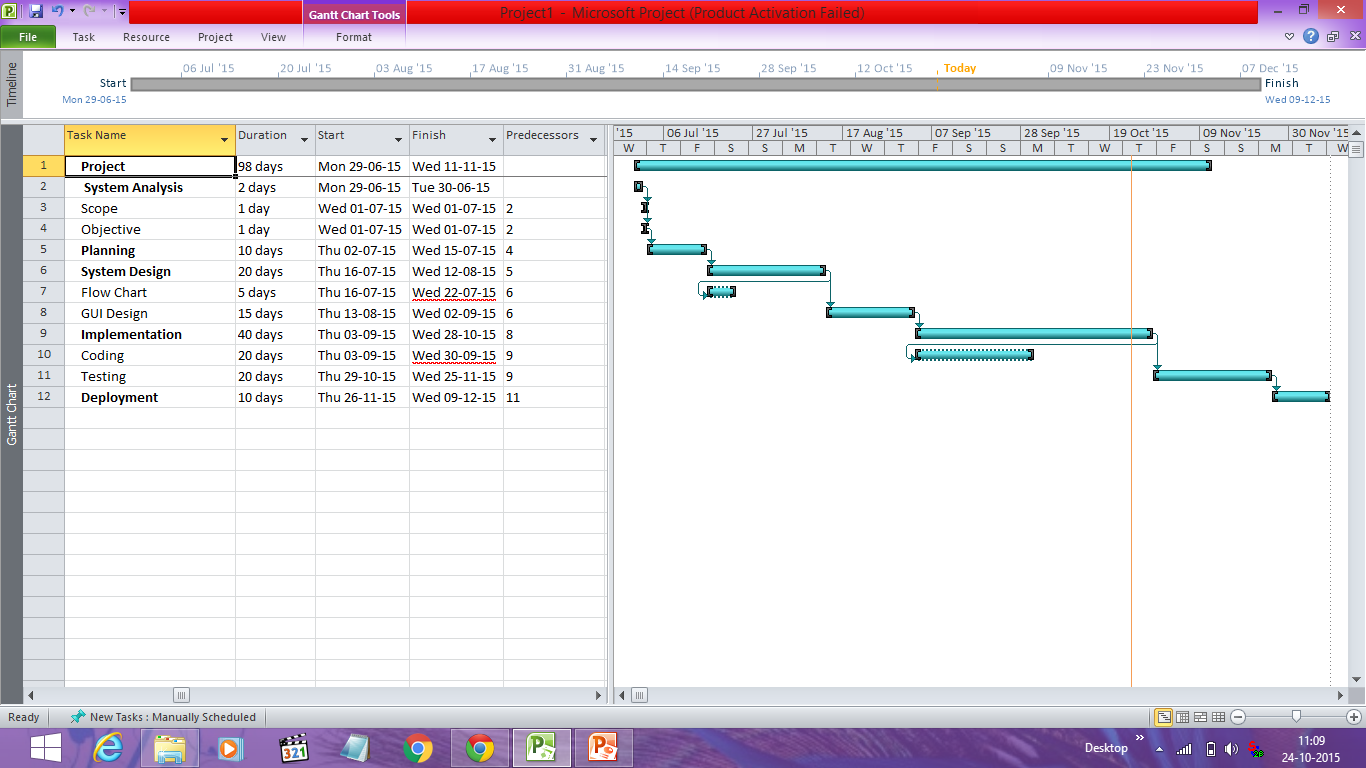
## API Level:

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform.

|  |  |  |
| --- | --- | --- |
| **Platform Version** | **API Level** | **VERSION\_CODE** |
| Android 5.1 | 22 | LOLLIPOP\_MR1 |
| Android 5.0 | 21 | LOLLIPOP |
| Android 4.4W | 20 | KITKAT\_WATCH |
| Android 4.4 | 19 | KITKAT |
| Android 4.3 | 18 | JELLY\_BEAN\_MR2 |
| Android 4.2, 4.2.2 | 17 | JELLY\_BEAN\_MR1 |
| Android 4.1, 4.1.1 | 16 | JELLY\_BEAN |
| Android 4.0.3, 4.0.4 | 15 | ICE\_CREAM\_SANDWICH\_MR1 |
| Android 4.0, 4.0.1, 4.0.2 | 14 | ICE\_CREAM\_SANDWICH |
| Android 3.2 | 13 | HONEYCOMB\_MR2 |
| Android 3.1.x | 12 | HONEYCOMB\_MR1 |
| Android 3.0.x | 11 | HONEYCOMB |
| Android 2.3.4  Android 2.3.3 | 10 | GINGERBREAD\_MR1 |
| Android 2.3.2  Android 2.3.1  Android 2.3 | 9 | GINGERBREAD |
| Android 2.2.x | 8 | FROYO |
| Android 2.1.x | 7 | ECLAIR\_MR1 |
| Android 2.0.1 | 6 | ECLAIR\_0\_1 |
| Android 2.0 | 5 | ECLAIR |
| Android 1.6 | 4 | DONUT |
| Android 1.5 | 3 | CUPCAKE |
| Android 1.1 | 2 | BASE\_1\_1 |
| Android 1.0 | 1 | BASE |

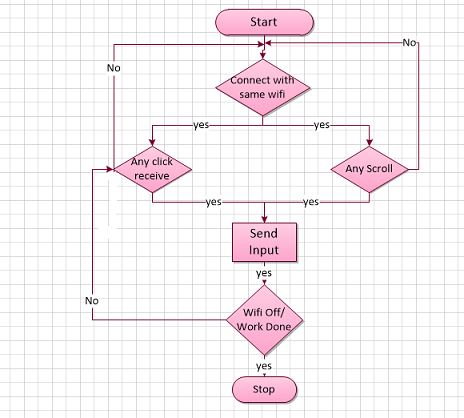
1. **System Design**

**3.1 Project Flow Planning:**

****

FIG\_4 Project Planning

**3.2 Flow Chart Design**

****

FIG\_5 Flow Chart

**3.3 GUI Design**

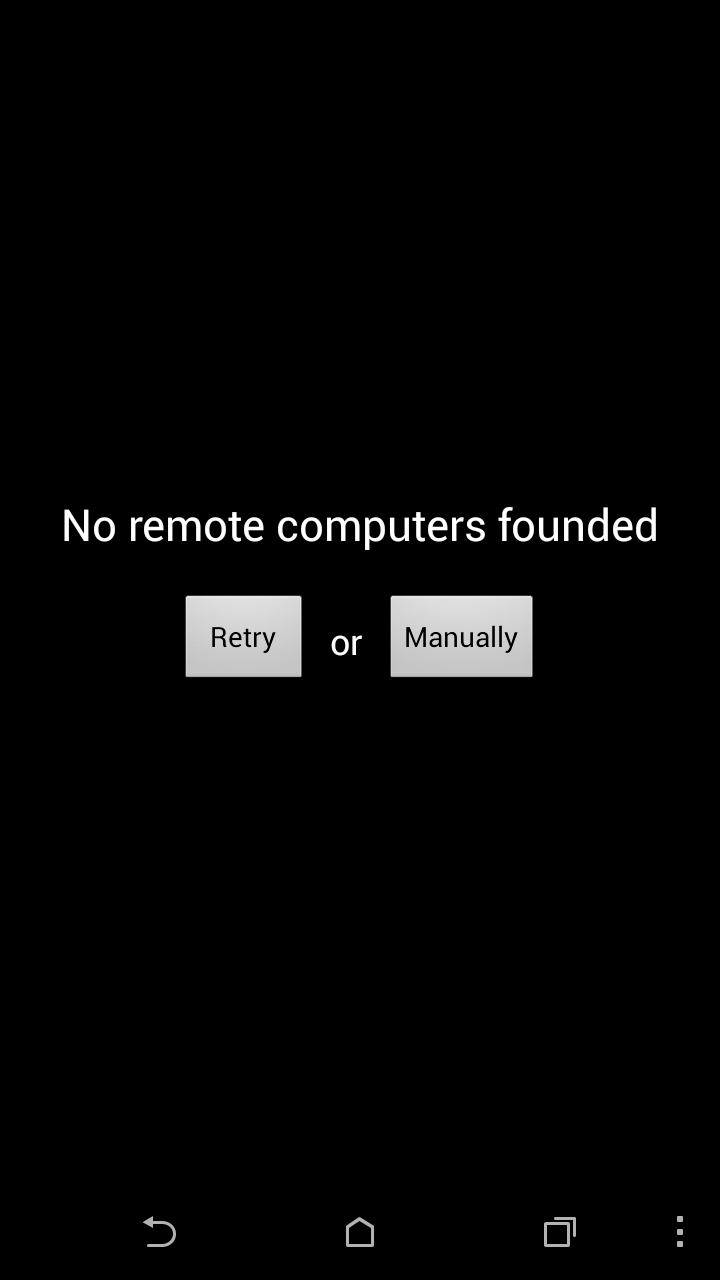
**1. Before entering the IP address:**

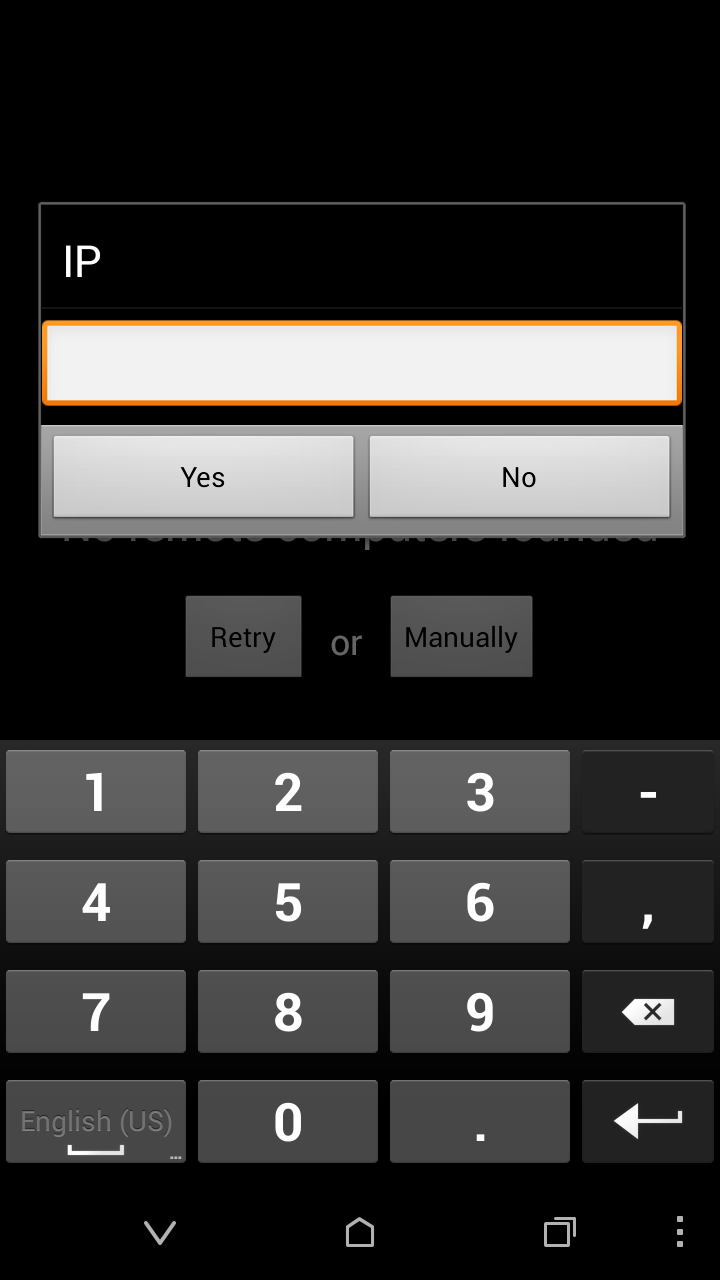
At the first step there is two ways of accepting the IP address

1. App will take IP address automatically.

Or

1. We have to enter IP Address of the laptop manually.

****

****

FIG\_6 Initial step

**2. After entering the IP address:**

After entering the IP address, app will get the control of all mouse activity via desktop server which we had initialized on laptop.

****

FIG\_7 Button off

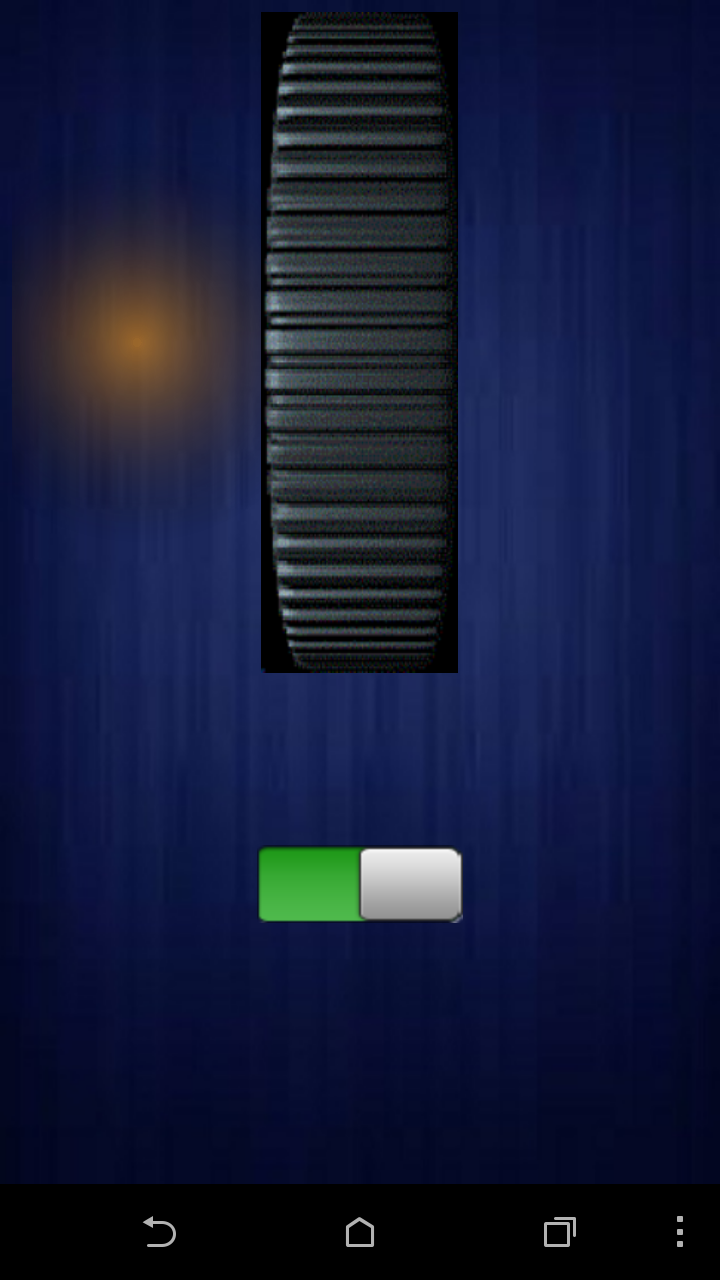
1. **Switch on when we need to work with the application:**

This switch will work as on and off for the application. When this switch is on we can work with the app and when this switch is off app will terminate the connection temporary.

****

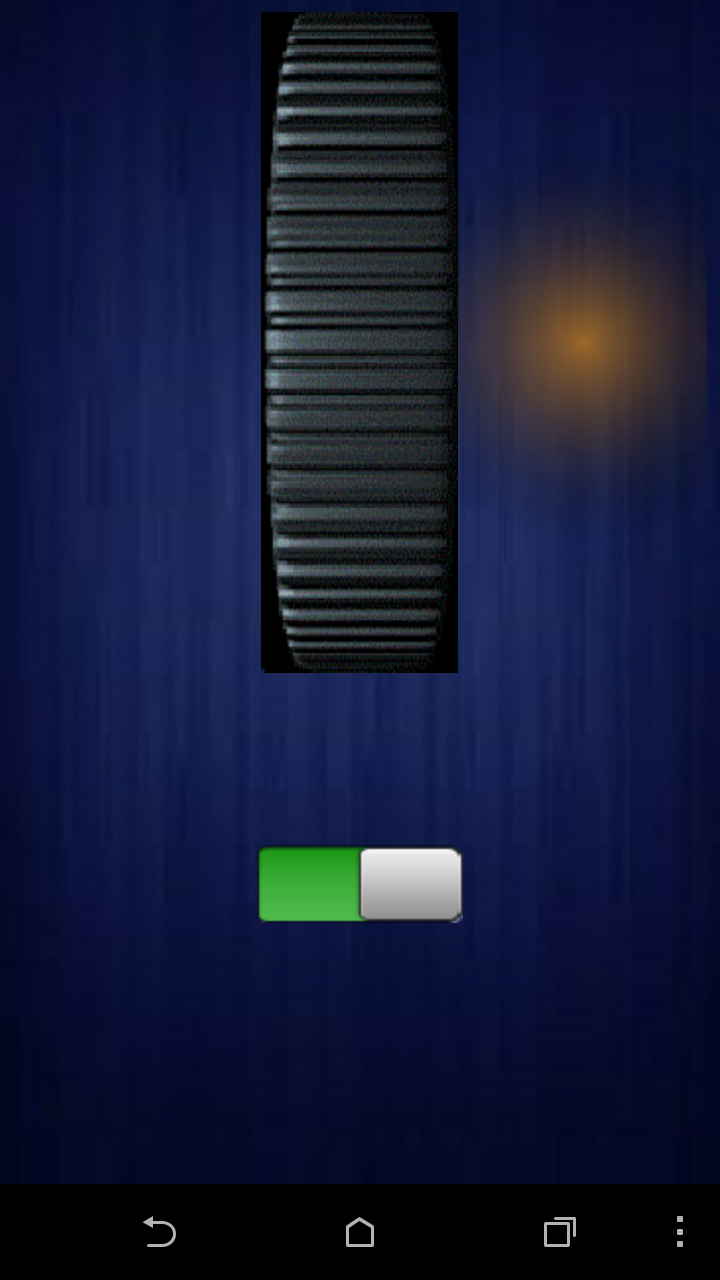
FIG\_8 Button On

1. **Left click and Right click**
2. **Left click event**

****

FIG\_9 Left Click

1. **Right Click Event:**

****

FIG\_10 Right Click

**3.4 Sample Code:**

**1. Sample Code of server:**

|  |
| --- |
| * **Connection Code:** |
| public class DiscoverTest { |
|  |
| /\*\* |
| \* @param args |
| \* @throws IOException |
| \*/ |
| public static void main(String[] args) throws IOException { |
| byte[] buf = new byte[2]; |
| InetAddress broadcastIP = Inet4Address.getByName("255.255.255.255"); |
|  |
| for (int i = 0; i < 10; i++) { |
| DatagramSocket socket = new DatagramSocket(); |
| socket.setSoTimeout(5000); |
| DatagramPacket packet = new DatagramPacket(buf, buf.length, broadcastIP, |
| Server.SERVER\_PORT); |
| buf[0] = Server.PACKET\_TYPE\_DISCOVER; |
| buf[1] = 1; |
| socket.send(packet); |
| socket.receive(packet); |
| if (packet.getLength() == 1) { |
| byte[] buffer = packet.getData(); |
| if (buffer[0] != 0x2) { |
| throw new Error(); |
| } |
| } else { |
| throw new Error(); |
| } |
| socket.close(); |
| } |
| System.out.println("Test finished."); |
| } |
|  |
| } |
| * **Initialization Code** |
|  |
| public void start() throws InterruptedException, AWTException { |
| STD\_OUT.println("Initializing..."); |
| init(); |
|  |
| STD\_OUT.println("Starting packet dispatcher..."); |
| PacketThread th; |
| try { |
| th = new PacketThread(); |
| th.start(); |
|  |
| STD\_OUT.println("The Air Mouse Server has started!"); |
|  |
| while (true) { |
| long lastTime = 0l; |
| MouseEvent event = mEvents.take(); |
| if (event.timestamp > lastTime) { |
| lastTime = event.timestamp; |
|  |
| switch (event.type) { |
| case MouseEvent.TYPE\_MOVE: |
| PointerInfo info = MouseInfo.getPointerInfo(); |
| if (info != null) { |
| Point point = info.getLocation(); |
| mRobot.mouseMove((int) (event.x \* mMaxMotionDist) + point.x, |
| (int) (event.y \* mMaxMotionDist) + point.y); |
| } |
| break; |
| case MouseEvent.TYPE\_PRESS: |
| mRobot.mousePress(event.button); |
| break; |
| case MouseEvent.TYPE\_RELEASE: |
| mRobot.mouseRelease(event.button); |
| break; |
| } |
| } |
| } |
| } catch (IOException e) { |
| STD\_OUT.println("Failed to launch the server. Is there already a running server?"); |
| } |
| } |
|  |
| private void printLog(String str) { |
| if (DEBUG\_MODE) { |
| STD\_OUT.print(str); |
| } |
| } |
|  |
| private void printlnLog(String str) { |
| if (DEBUG\_MODE) { |
| STD\_OUT.println(str); |
| } |
| } |
|  |
| public static int convertButtonMask(int button) { |
| switch (button) { |
| case PACKET\_MOUSE\_BUTTON\_LEFT: |
| return InputEvent.BUTTON1\_MASK; |
| case PACKET\_MOUSE\_BUTTON\_RIGHT: |
| return InputEvent.BUTTON3\_MASK; |
| case PACKET\_MOUSE\_BUTTON\_MIDDLE: |
| return InputEvent.BUTTON3\_DOWN\_MASK; |
| default: |
| return -1; |
| } |
| } |
|  |

1. **Xml File Code:**

|  |
| --- |
| * **Mouse.xml** |
| <?xml version="1.0" encoding="utf-8"?> |
| <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" |
| android:id="@+id/layout\_mouse" |
| android:layout\_width="fill\_parent" |
| android:layout\_height="fill\_parent" |
| android:background="@drawable/bg" |
| android:orientation="vertical" > |
|  |
| <LinearLayout |
| android:layout\_width="match\_parent" |
| android:layout\_height="wrap\_content" |
| android:layout\_weight="2" |
| android:padding="6dp" > |
|  |
| <Button |
| android:id="@+id/btn\_mouse\_left" |
| android:background="@drawable/btn\_mouse" |
| android:layout\_width="wrap\_content" |
| android:layout\_height="match\_parent" |
| android:layout\_weight="3" /> |
|  |
| <ImageView |
| android:id="@+id/btn\_mouse\_wheel" |
| android:layout\_width="0dp" |
| android:layout\_height="match\_parent" |
| android:layout\_weight="1.5" |
| android:scaleType="fitXY" |
| android:src="@drawable/wheel" /> |
|  |
| <Button |
| android:id="@+id/btn\_mouse\_right" |
| android:background="@drawable/btn\_mouse" |
| android:layout\_width="wrap\_content" |
| android:layout\_height="match\_parent" |
| android:layout\_weight="1.8" /> |
| </LinearLayout> |
|  |
| <RelativeLayout |
| android:layout\_width="match\_parent" |
| android:layout\_height="wrap\_content" |
| android:layout\_weight="2" |
| android:gravity="center\_horizontal" > |
|  |
| <ToggleButton |
| android:id="@+id/tb\_switch" |
| android:layout\_width="wrap\_content" |
| android:layout\_height="wrap\_content" |
| android:layout\_marginTop="80dp" |
| android:background="@drawable/btn\_switch" |
| android:height="0dp" |
| android:textOff="" |
| android:textOn="" |
| android:width="0dp" /> |
| </RelativeLayout> |
|  |
| </LinearLayout> |

1. **Functions Of main activity:**

|  |
| --- |
| 1. **private void** onMoveCommand(**float** x, **float** y) {  **if** (**mServerAddr** != **null**) {  **mPacketBuffer**.clear();  **mPacketBuffer**.put(***PACKET\_TYPE\_MOVE***);  **mPacketBuffer**.putLong(System.*currentTimeMillis*());  **mPacketBuffer**.putFloat(x);  **mPacketBuffer**.putFloat(y);  **mPacketBuffer**.flip();   sendPacket();  } |
| 2. **private void** onPressCommand(**int** button) {  **if** (**mServerAddr** != **null**) {  **mPacketBuffer**.clear();  **mPacketBuffer**.put(***PACKET\_TYPE\_PRESS***);  **mPacketBuffer**.putLong(System.*currentTimeMillis*());  **mPacketBuffer**.putInt(button);  **mPacketBuffer**.flip();   **for** (**int** i = 0; i < 3; i++) {  sendPacket();  **mPacketBuffer**.rewind();  }  } } |
| 3. **private void** onReleaseCommand(**int** button) {  **if** (**mServerAddr** != **null**) {  **mPacketBuffer**.clear();  **mPacketBuffer**.put(***PACKET\_TYPE\_RELEASE***);  **mPacketBuffer**.putLong(System.*currentTimeMillis*());  **mPacketBuffer**.putInt(button);  **mPacketBuffer**.flip();   **for** (**int** i = 0; i < 3; i++) {  sendPacket();  **mPacketBuffer**.rewind();  }  } } |
| 1. **private void** sendPacket() {  **if** (**mServerAddr** != **null** && !**mPausing**) {  **try** {  **mSocketChannel**.send(**mPacketBuffer**, **mServerAddr**);  } **catch** (IOException e) {  *//* ***TODO Auto-generated catch block*** e.printStackTrace();  }  } } |
| 1. **protected** Void doInBackground(Void... params) {  *// Test mode  // - specified server address* **if** (***TEST\_MODE***) {  **mServerAddr** = **new** InetSocketAddress(***TEST\_SERVER\_IP***, ***SERVER\_PORT***);  **return null**;  }   **byte**[] buf = **new byte**[2];  InetAddress broadcastIP;  **try** {  broadcastIP = Inet4Address.*getByName*(**"255.255.255.255"**);  **try** {  DatagramSocket socket = **new** DatagramSocket();  socket.setSoTimeout(8000);   DatagramPacket packet = **new** DatagramPacket(buf, buf.**length**, broadcastIP,  ***SERVER\_PORT***);  buf[0] = ***PACKET\_TYPE\_DISCOVER***;  buf[1] = ***PROTOCAL\_VERSION***;  **for** (**int** i = 0; i < 3; i++) {  socket.send(packet);  }  socket.receive(packet);  **if** (packet.getLength() == 1) {  **byte**[] buffer = packet.getData();  **if** (buffer[0] != ***PACKET\_TYPE\_REPLY***) {  **mServerAddr** = **null**;  **return null**;  }  } **else** {  **mServerAddr** = **null**;  **return null**;  }  **mServerAddr** = packet.getSocketAddress();  socket.close();  } **catch** (IOException e) {  e.printStackTrace();  }  } **catch** (UnknownHostException e1) {  e1.printStackTrace();  }  **return null**;  } } |
| 1. **private void** chooseSensor() {  **mSensor** = **mSensorManager**.getDefaultSensor(Sensor.***TYPE\_GRAVITY***);  **mEpsilon** = ***EPSILON\_GRAVITY***;  **if** (**mSensor** == **null**) {  **mSensor** = **mSensorManager**.getDefaultSensor(Sensor.***TYPE\_ACCELEROMETER***);  **mEpsilon** = ***EPSILON\_ACCELEROMETER***;  } } |
| 1. **public void** onAccuracyChanged(Sensor sensor, **int** accuracy) { }  @Override **public void** onSensorChanged(SensorEvent event) {  **boolean** valuesUpdated = **false**;  **float** values[] = event.**values**;    **float** newX = -values[0];    **float** newY = -values[1];   **if** (**mTbSwitch**.isChecked()) {   **float** deltaX = 0f;  **float** deltaY = 0f;  **if** (Math.*abs*(newX - **mLastX**) > **mEpsilon**) {  deltaX = newX - **mLastX**;  **mLastX** = newX;  valuesUpdated = **true**;  }  **if** (Math.*abs*(newY - **mLastY**) > **mEpsilon**) {  deltaY = newY - **mLastY**;  **mLastY** = newY;  valuesUpdated = **true**;  }  **if** (valuesUpdated) {  **float** pos[] = {  deltaX,  deltaY };  Message msg = **mSocketHandler**.obtainMessage(***MSG\_MOVE\_COMMAND***, pos);  **mSocketHandler**.sendMessage(msg);  }  } **else** {  **mLastX** = newX;  **mLastY** = newY;  } } |

1. **Limitations and Future Enhancement**

* **Limitation**
* Need android phone to run application
* Both device must connected in same Wi-Fi
* we can’t use all the gesture of mouse
* connection depends upon w-fi range
* We have to make application manually for all the users because we have to add IP address of particular user’s laptop.
* **Future Enhancement**
* We can add keyboard in our application.
* Make it available for i-pone, black-berry and windows.

1. **Conclusion**

Till now we have discussed in brief about the project of Android App to use smartphone as mouse using sensor data processing. We have discussed some of its benifites,limitations, requirements etc. The ultimate goal of this App is to solve majority of the problems like carrying hardware devises with us everytime,hardware failure,etc.We can easily use this application. It also have some limitations but we can use this app temporarily.This app also have some future enhancement like we can add keyboard in this application.

1. **References:**

<http://en.wikibooks.org/wiki/Sensor>   
<http://www.howstuffworks.com/>phone-mouse  
[http://www.google.co.in](http://www.google.co.in/)/GIThub  
[http://www.library.thinkquest.org](http://www.library.thinkquest.org/)