Manisha Vyas

Step 1:

Read on Socket Programming. (Client - Server Connection) and Telnet connection.

Step 2:

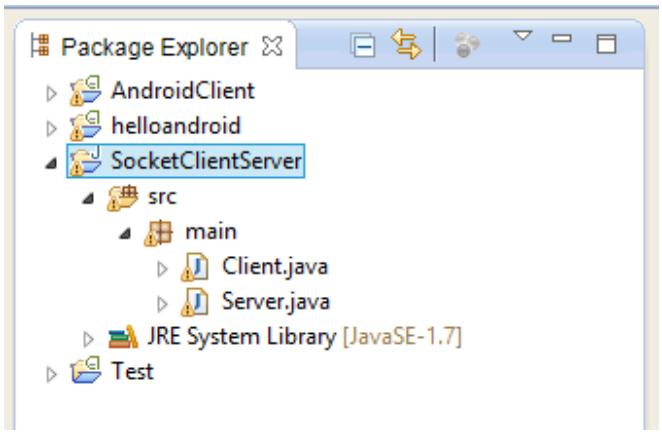
Read on how to connect
Download Google Sensor Simulator from

https://code.google.com/p/openintents/wiki/SensorSimulator

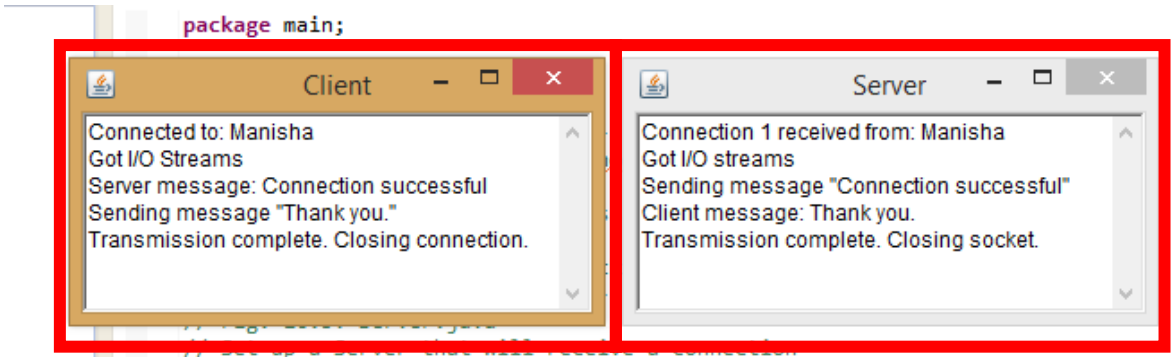
Step3:

Q3_SocketClientServer:: Client - Server Simple Java Program to check connection.

- You should make your programs work even if wired or wireless network is available.



Run Server program.... Then run Client Program.

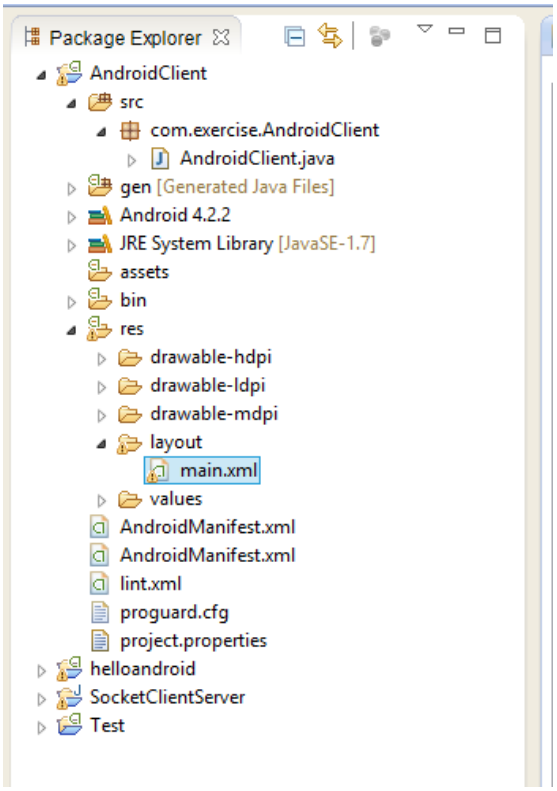


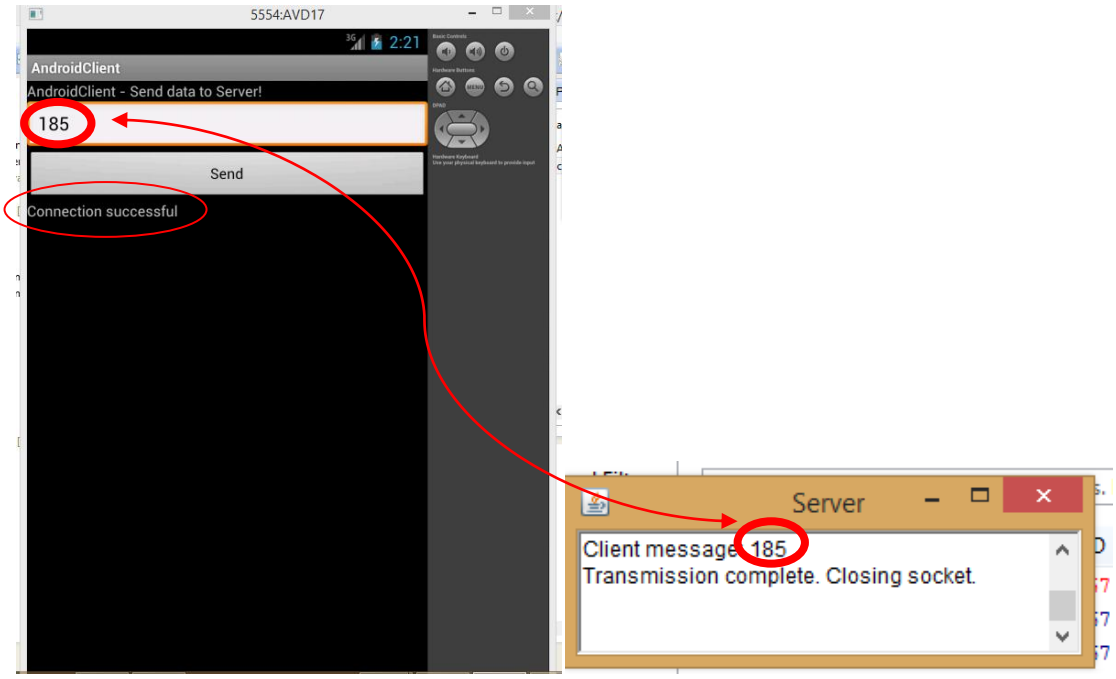
Step4:

Q4_AndroidClient:: Socket programming between an Android client and a PC server

- This is the sequence of steps for sending of message from the Android to the server.
  a. The user enter a message on the textfield.
  b. The user presses a button to send the entered message from the Android to the server.

Keep the server ready.. Start Android application.





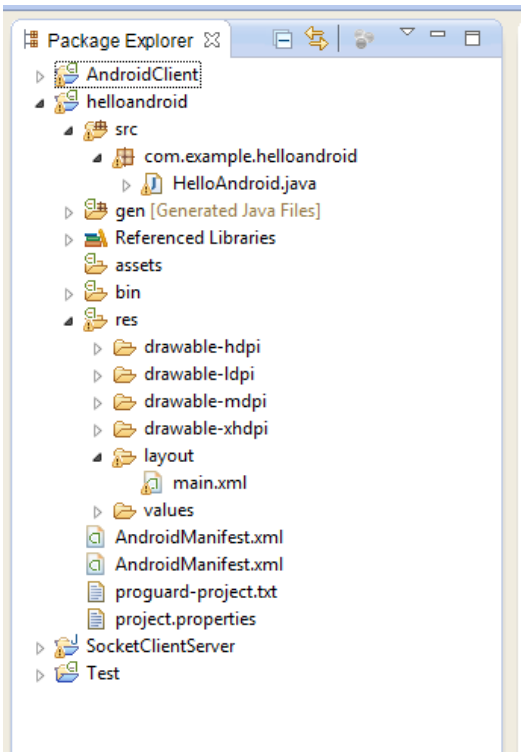
Step5:

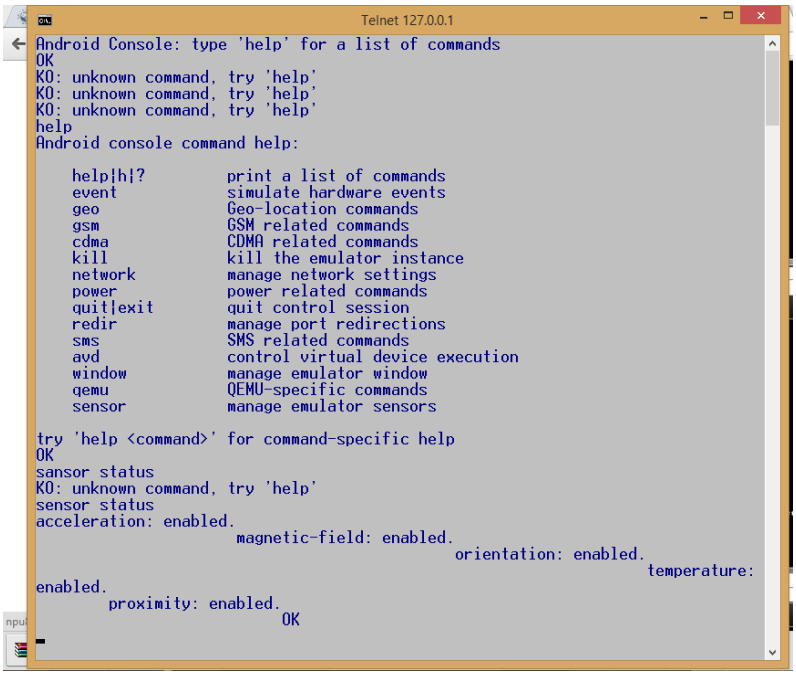
Q5_helloandroid :: Use Telnet to get into Android Shell to change Accelerometer and Gyroscope/Orientation data on an Android Emulator -- Professor Adam Weng
Note:

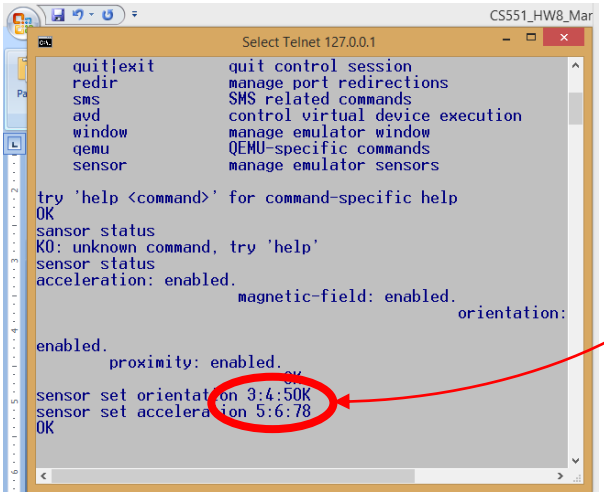
Replace Gyroscope with the deprecated Orientation if the emulator does not support Gyroscope.

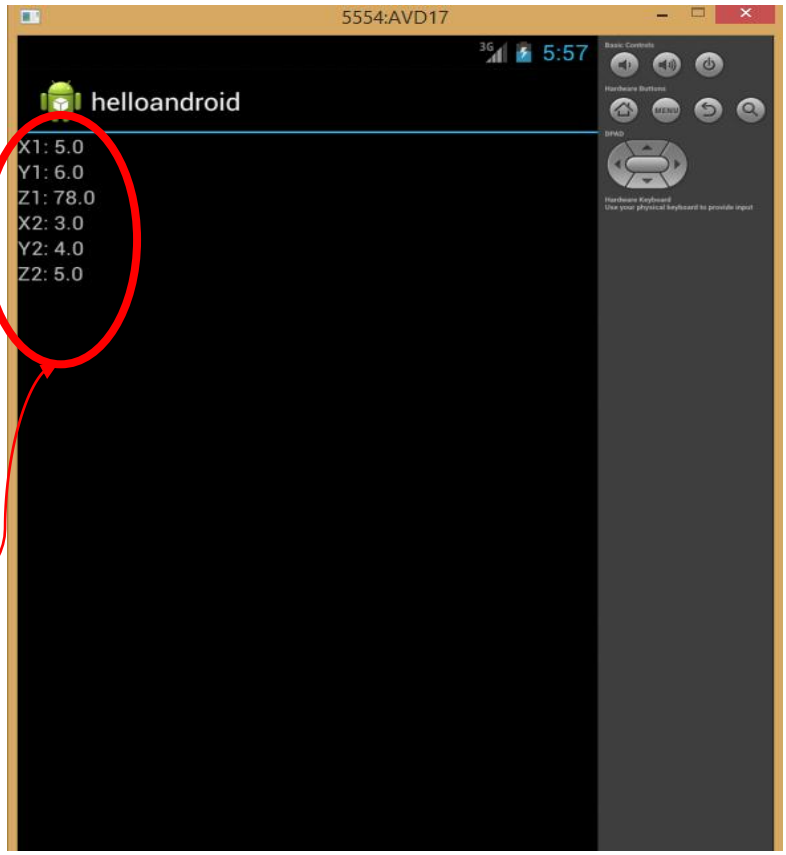
integrate sensor simulator with your code so that the sensor data is obtained from sensor simulator.

- This algorithm uses only the three coordinates of the accelerometer to calculate G Force to find the risk class. First the vector sum is calculated (Round(pow(x, 2) + pow(y, 2)+ pow(z, 2))). The risk class is calculated based on this value. The following table shows the value
- A message will be sent back to the patient's device based on the risk class.
- Another approach is to Use G-force to determine whether one falls on the floor.


Package Explorer
AndroidClient
helloandroid
src
com.example.helloandroid
HelloAndroid.java
gen [Generated Java Files]
Referenced Libraries
assets
bin
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
drawable-xhdpi
layout
main.xml
values
AndroidManifest.xml
AndroidManifest.xml
proguard-project.txt
project.properties
SocketClientServer
Test


Telnet 127.0.0.1
Android Console: type 'help' for a list of commands
OK
KO: unknown command, try 'help'
KO: unknown command, try 'help'
KO: unknown command, try 'help'
help
Android console command help:
help|h|?
print a list of commands
event
simulate hardware events
geo
Geo-location commands
gsm
GSM related commands
cdma
CDMA related commands
kill
kill the emulator instance
network
manage network settings
power
power related commands
quit|exit
quit control session
redir
manage port redirections
sms
SMS related commands
avd
control virtual device execution
window
manage emulator window
qemu
QEMU-specific commands
sensor
manage emulator sensors
try 'help <command>' for command-specific help
OK
sansor status
KO: unknown command, try 'help'
sensor status
acceleration: enabled.
magnetic-field: enabled.
orientation: enabled.
temperature:
enabled.
proximity: enabled.
OK


CS551_HW8_Mar
Select Telnet 127.0.0.1
quit|exit
quit control session
redir
manage port redirections
sms
SMS related commands
avd
control virtual device execution
window
manage emulator window
qemu
QEMU-specific commands
sensor
manage emulator sensors
try 'help <command>' for command-specific help
OK
sansor status
KO: unknown command, try 'help'
sensor status
acceleration: enabled.
magnetic-field: enabled.
orientation:
enabled.
proximity: enabled.
sensor set orientation 3:4:5OK
sensor set acceleration 5:6:78
OK

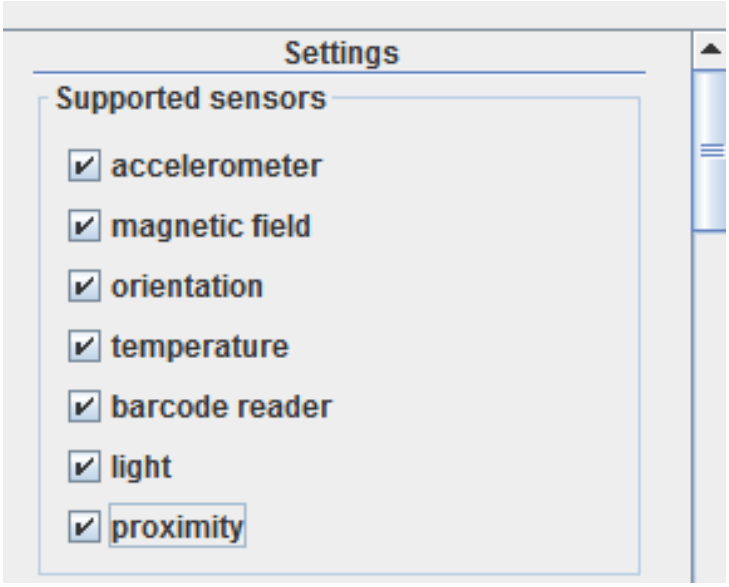

5554:AVD17
5:57
helloandroid
X1: 5.0
Y1: 6.0
Z1: 78.0
X2: 3.0
Y2: 4.0
Z2: 5.0

Step6:

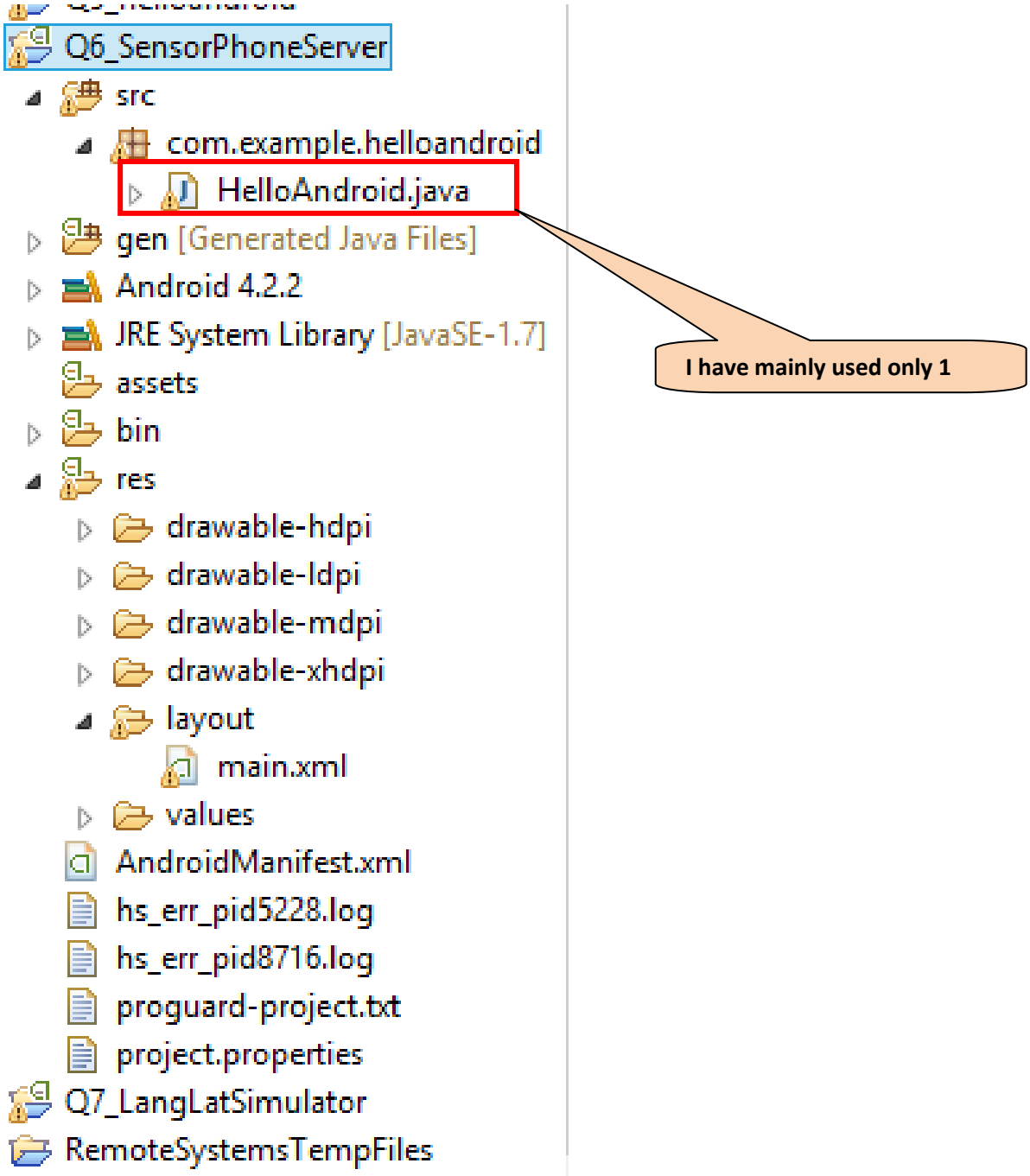
Q6_SensorPhoneServer::

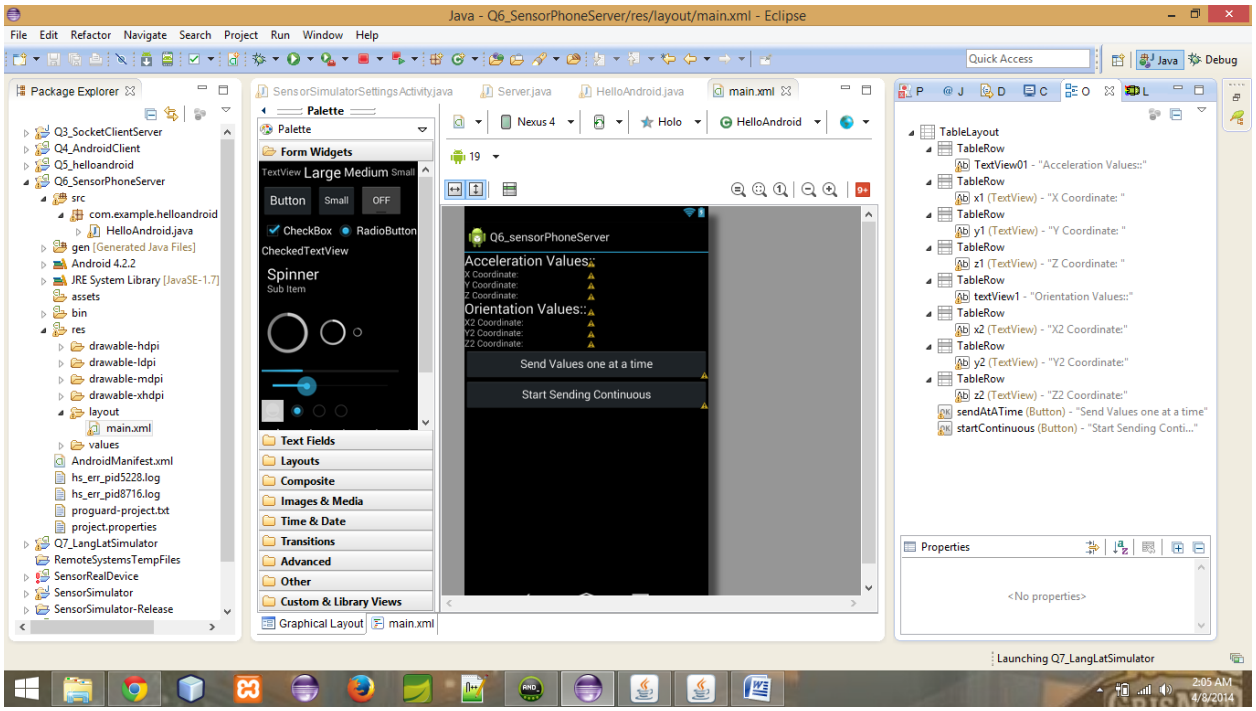
1. Change Geo-location data
    - Except for Accelerometer data and Gyroscope/Orientation/Orientation data, please also display Geo-location data on the Android's screen.
    - Except for Accelerometer data and Gyroscope/Orientation data, please also send Geo-location data from the Android client to the PC server.
    - integrate sensor simulator with your code so that the sensor data is obtained from sensor simulator.

I have done ALL the type of sensors (in step 6 implementation only) ... to show that we can enable 7 sensors from sensor Simulator and can get and send values to Server



program 7 uses manual sending data while program 8 is using Sensor Simulator

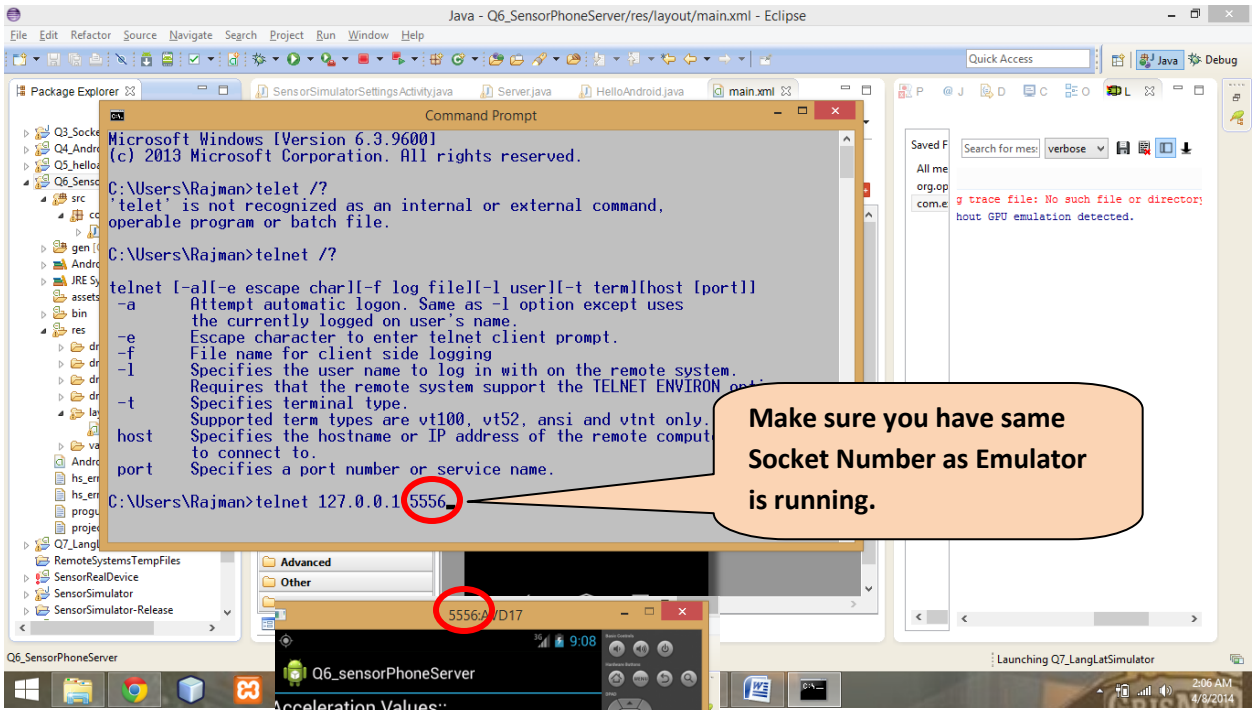

Q6_SensorPhoneServer
src
com.example.helloandroid
HelloAndroid.java
gen [Generated Java Files]
Android 4.2.2
JRE System Library [JavaSE-1.7]
assets
bin
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
drawable-xhdpi
layout
main.xml
values
AndroidManifest.xml
hs_err_pid5228.log
hs_err_pid8716.log
proguard-project.txt
project.properties
Q7_LangLatSimulator
RemoteSystemsTempFiles
I have mainly used only 1

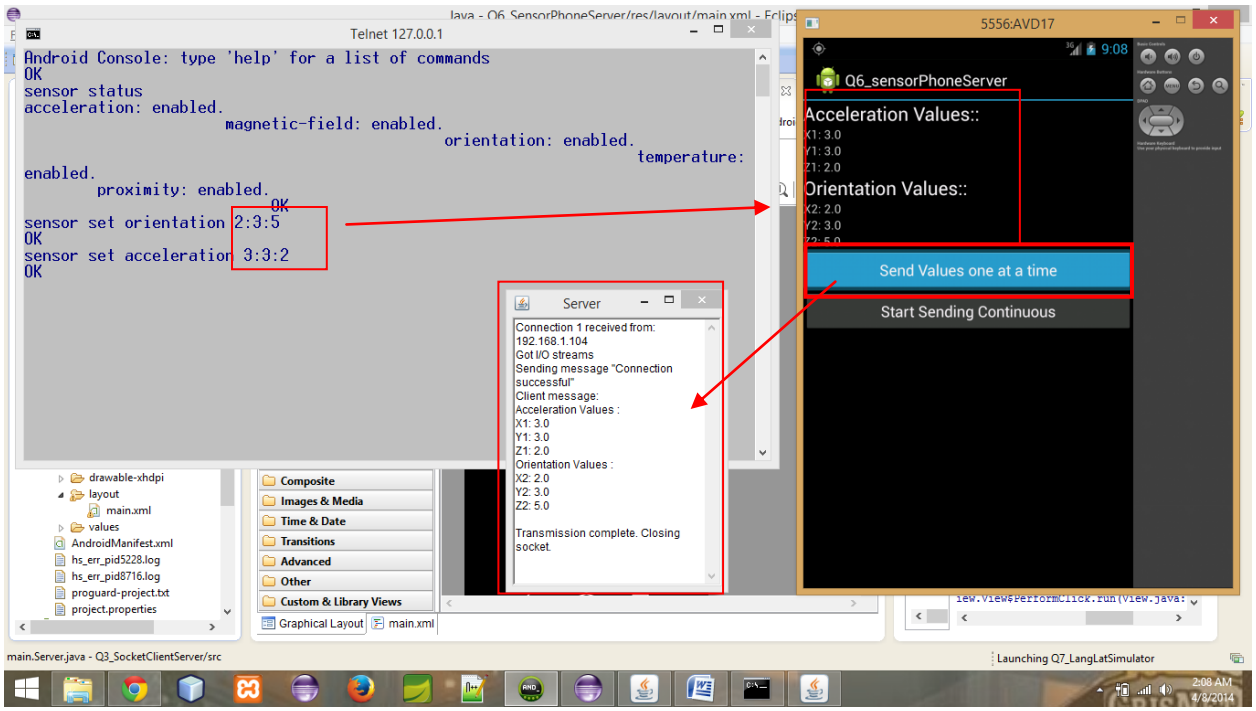


I have used 6 TextViews to Show Data

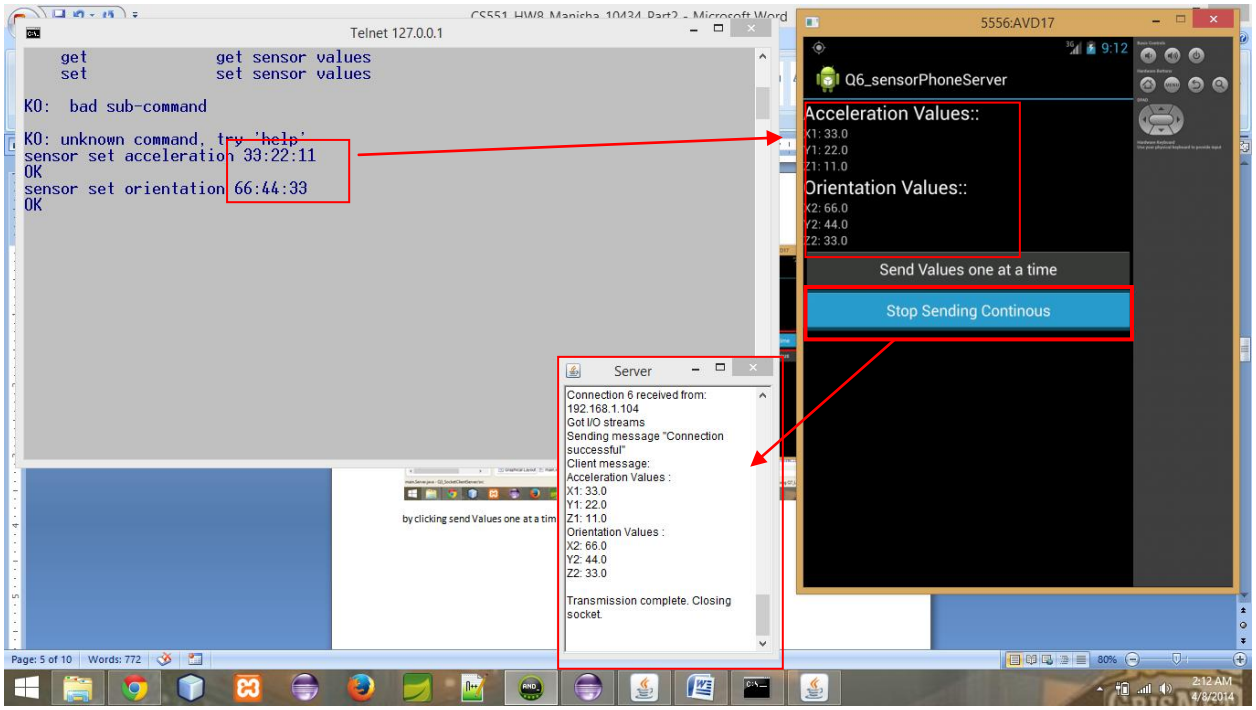
Output Screen Shots::

Connect telnet -> telnet 127.0.0.1 5556





by clicking send Values one at a time. Will only send one Value to server on One click.



By clicking Start Sending Continous button it will continously update Server if any changes made in values by Telnet.

HelloAdroid.java

```
package com.example.helloandroid;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;
import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.os.StrictMode;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class HelloAndroid extends Activity implements SensorEventListener,Runnable {
	private SensorManager sensorManager;

	TextView x1; // declare X axis object
	TextView y1; // declare Y axis object
	TextView z1; // declare Z axis object

	TextView x2; // declare X axis object
	TextView y2; // declare Y axis object
	TextView z2; // declare Z axis object

	String x1Str,y1Str,z1Str,x2Str,y2Str,z2Str ;
	String oldX1,oldY1,oldZ1,oldX2,oldY2,oldZ2;

	Button sendAtATime,startContinous,dataChanged;
	private boolean startStop = false ,valueChanged = true;

	public HelloAndroid(){}

	public HelloAndroid(String x1Str, String y1Str, String z1Str, String x2Str,
			String y2Str, String z2Str) {
		super();
		this.x1Str = x1Str;
		this.y1Str = y1Str;
		this.z1Str = z1Str;
		this.x2Str = x2Str;
		this.y2Str = y2Str;
		this.z2Str = z2Str;
	}

	@Override
	public void onCreate(Bundle savedInstanceState){

		StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder()
```

Made Thread

```
            .detectDiskReads()
            .detectDiskWrites()
            .detectNetwork()   // or .detectAll() for all detectable problems
            .penaltyLog()
            .build());
          StrictMode.setVmPolicy(new StrictMode.VmPolicy.Builder()
            .detectLeakedSqlLiteObjects()
            .detectLeakedClosableObjects()
            .penaltyLog()
            .penaltyDeath()
            .build());
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);

          x1=(TextView)findViewById(R.id.x1); // create X axis object
          y1=(TextView)findViewById(R.id.y1); // create Y axis object
          z1=(TextView)findViewById(R.id.z1); // create Z axis object

          x2=(TextView)findViewById(R.id.x2); // create X axis object
          y2=(TextView)findViewById(R.id.y2); // create Y axis object
          z2=(TextView)findViewById(R.id.z2); // create Z axis object

          sendAtATime =  (Button)findViewById(R.id.sendAtATime);
          startContinous =  (Button)findViewById(R.id.startContinuous);

        sendAtATime.setOnClickListener(buttonSendOnClickListener);
        startContinous.setOnClickListener(buttonContinuousClickListener);

        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
          // add listener. The listener will be HelloAndroid (this) class
          sensorManager.registerListener(this,
                      sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                      SensorManager.SENSOR_DELAY_NORMAL);

          sensorManager.registerListener(this,
                      sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
                      SensorManager.SENSOR_DELAY_NORMAL);
    }

    public void onAccuracyChanged(Sensor sensor,int accuracy){
    }

    public void onSensorChanged(SensorEvent event)
    {
          // check sensor type
          if(event.sensor.getType()==Sensor.TYPE_ACCELEROMETER)
          {
                oldX1 = x1.getText().toString();
                oldY1 = y1.getText().toString();
                oldZ1 = z1.getText().toString();

                // assign directions/
                float x=event.values[0];
                float y=event.values[1];
                float z=event.values[2];
```

this method continuously checks for any changes made on sensor

```
                x1.setText("X1: "+x);
                y1.setText("Y1: "+y);
                z1.setText("Z1: "+z);

            }
            if(event.sensor.getType()==Sensor.TYPE_ORIENTATION)
            {
                oldX2 = x2.getText().toString();
                oldY2 = y2.getText().toString();
                oldZ2 = z2.getText().toString();

                // assign directions/
                float x=event.values[0];
                float y=event.values[1];
                float z=event.values[2];

                x2.setText("X2: "+x);
                y2.setText("Y2: "+y);
                z2.setText("Z2: "+z);
            }

            if(x1.getText().toString().equals(oldX1) &&
                    y1.getText().toString().equals(oldY1)
            && z1.getText().toString().equals(oldZ1) &&
                    x2.getText().toString().equals(oldX2)
            && y2.getText().toString().equals(oldY2) &&
                    z2.getText().toString().equals(oldZ2) )
            {
                valueChanged = false;
            }
            else
            {
                valueChanged = true;
            }
            if(startStop && valueChanged)
            {

            Thread aThread = new Thread(new HelloAndroid(x1.getText().toString()
                                                    ,y1.getText().toString()
                                                    ,z1.getText().toString()
                                                    ,x2.getText().toString()
                                                    ,y2.getText().toString()
                                                    ,z2.getText().toString()));
                aThread.run();
            }
        }

        Button.OnClickListener buttonContinuousClickListener = new
Button.OnClickListener()
```

Creating Thread so that we can run Sensor (Telnet) **and Server Connection together**

```
	{
		public void onClick(View arg0)
		{
			if(startStop)
			{
				startStop = false;
				startContinous.setText("Start Sending Continous");
				return;
			}
			startStop = true;
			startContinous.setText("Stop Sending Continous");
		}
	 };
	 Button.OnClickListener buttonSendOnClickListener = new
Button.OnClickListener()
	 {
		public void onClick(View arg0)
		{
			Thread aThread = new Thread(new
HelloAndroid(x1.getText().toString()

	,y1.getText().toString()

	,z1.getText().toString()

	,x2.getText().toString()

	,y2.getText().toString()

	,z2.getText().toString()));
			aThread.run();
		}
	 };
	public void run()
	{
		Socket socket = null;
		 DataOutputStream dataOutputStream = null;
		 DataInputStream dataInputStream = null;

		 try
		 {
			socket = new Socket("192.168.1.104", 5000);
			dataOutputStream = new
DataOutputStream(socket.getOutputStream());
			dataInputStream = new DataInputStream(socket.getInputStream());
			dataOutputStream.writeUTF("\nAcceleration Values :\n"
								+x1Str+"\n"
								+y1Str+"\n"
								+z1Str+"\n"
								+"Orientation
Values :\n"
								+x2Str+"\n"
								+y2Str+"\n"
								+z2Str+"\n");
		 }
```

Thread, to connect to Server and send Values

```
            catch (UnknownHostException e)
            {
                  e.printStackTrace();
            }
            catch (IOException e)
            {
                  e.printStackTrace();
            }
            finally
            {
                        if (socket != null)
                        {
                              try
                              {
                                    socket.close();
                              }
                              catch (IOException e)
                              {
                                    e.printStackTrace();
                              }
                        }
                        if (dataOutputStream != null)
                        {
                               try
                               {
                                 dataOutputStream.close();
                               }
                               catch (IOException e)
                               {
                                      e.printStackTrace();
                               }
                        }
                        if (dataInputStream != null)
                        {
                              try
                              {
                                     dataInputStream.close();
                              }
                              catch (IOException e)
                              {
                                     e.printStackTrace();
                              }
                        }
                  }
      }
}
```

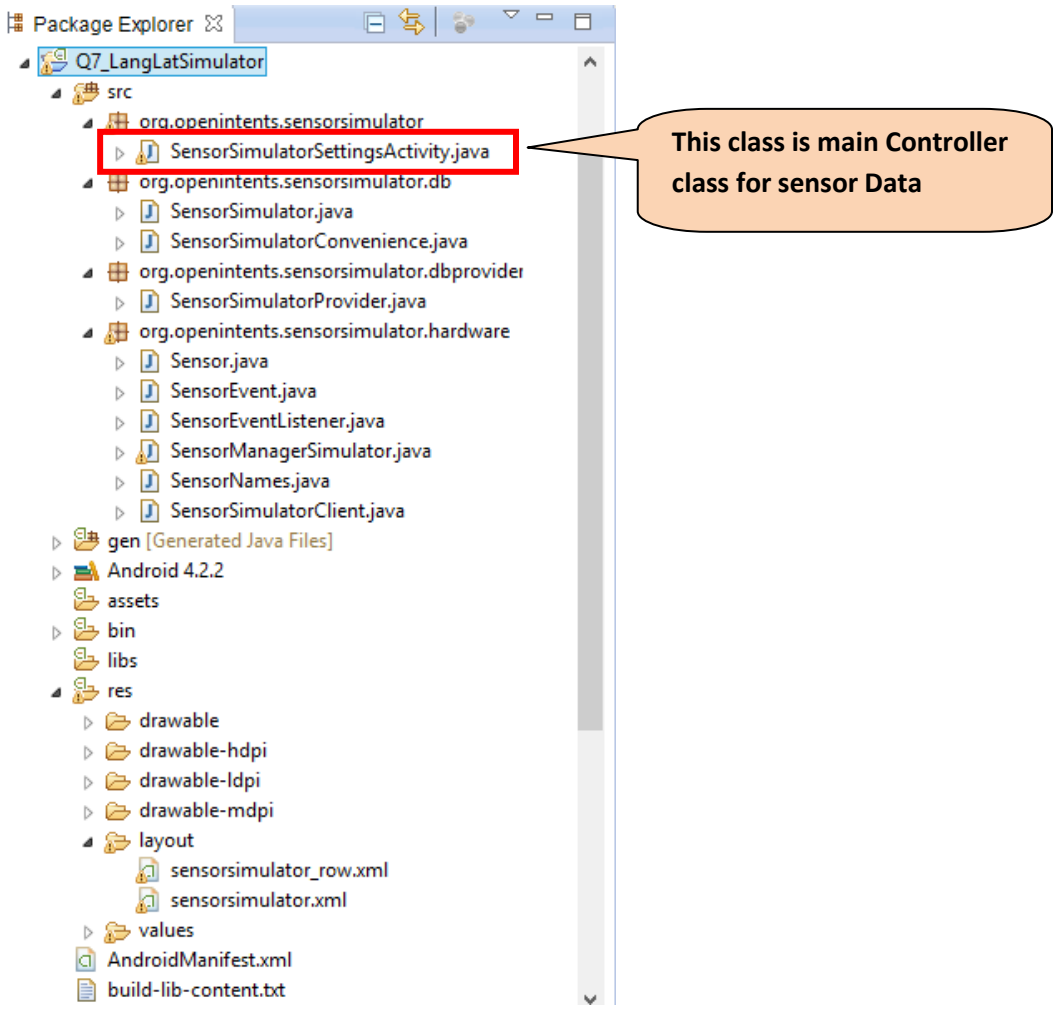
Step 7:

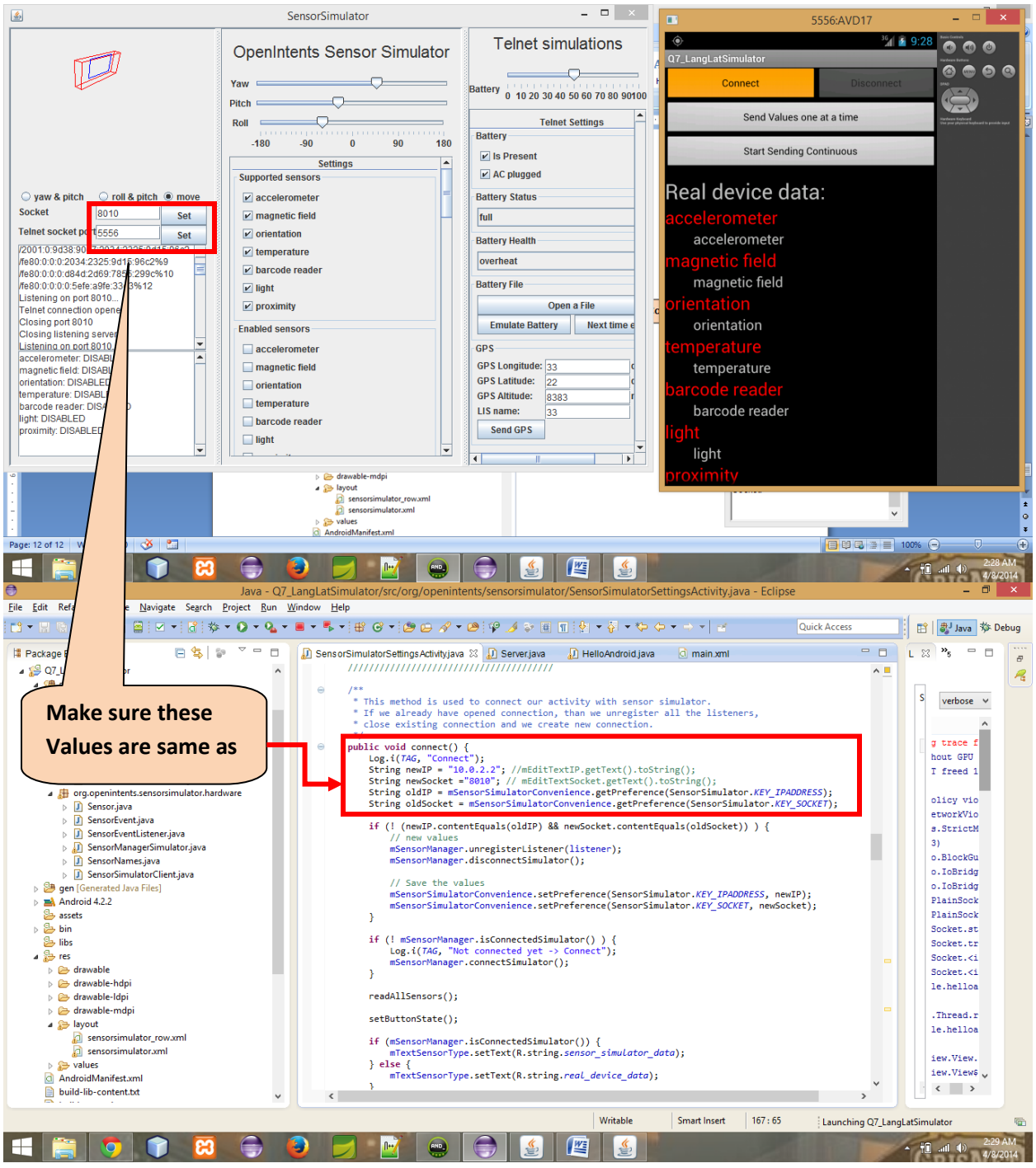
Q7_LangLatSimulator ::

Change Geo-location data

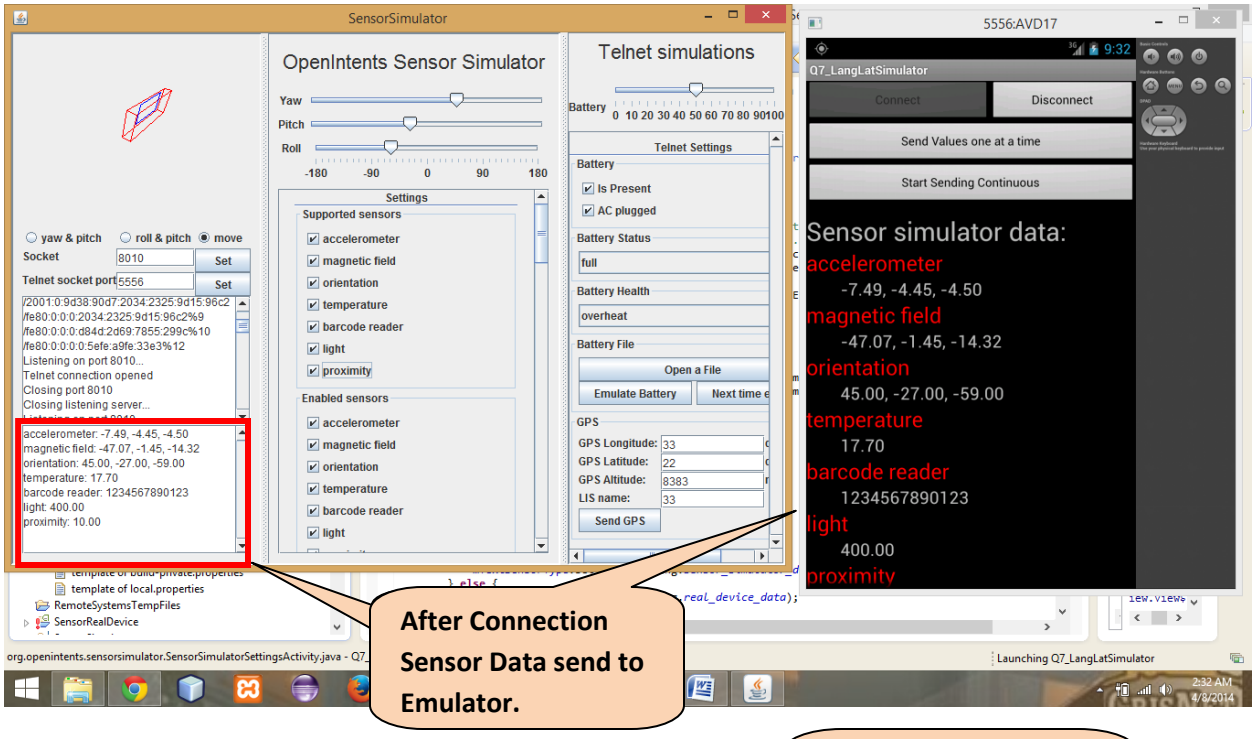
In this Android Program I have mainly used Sensor Simulator to connect with Android Emulator and which then sends data to Server.

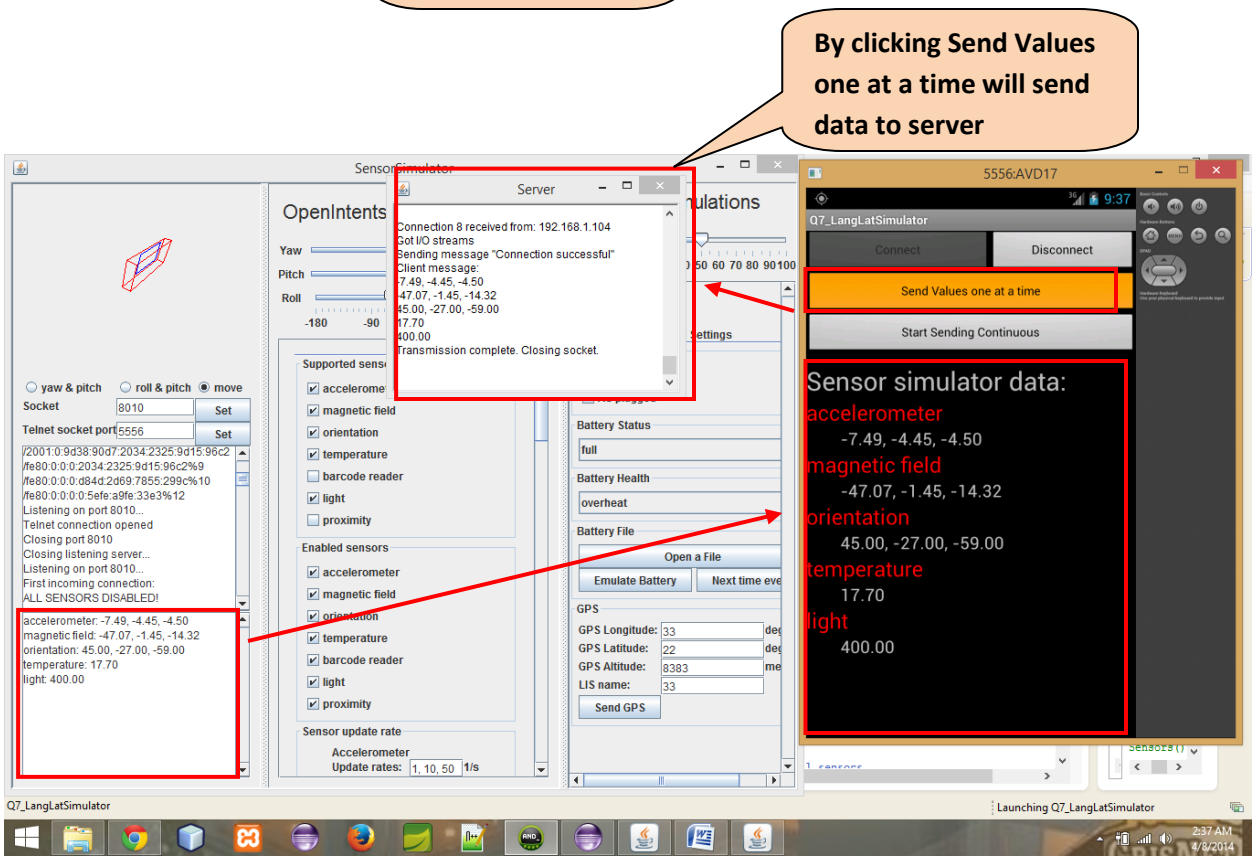
By using help of Android Sensor Simulator Sample Program i have mainly used java classes to connect to simulator.

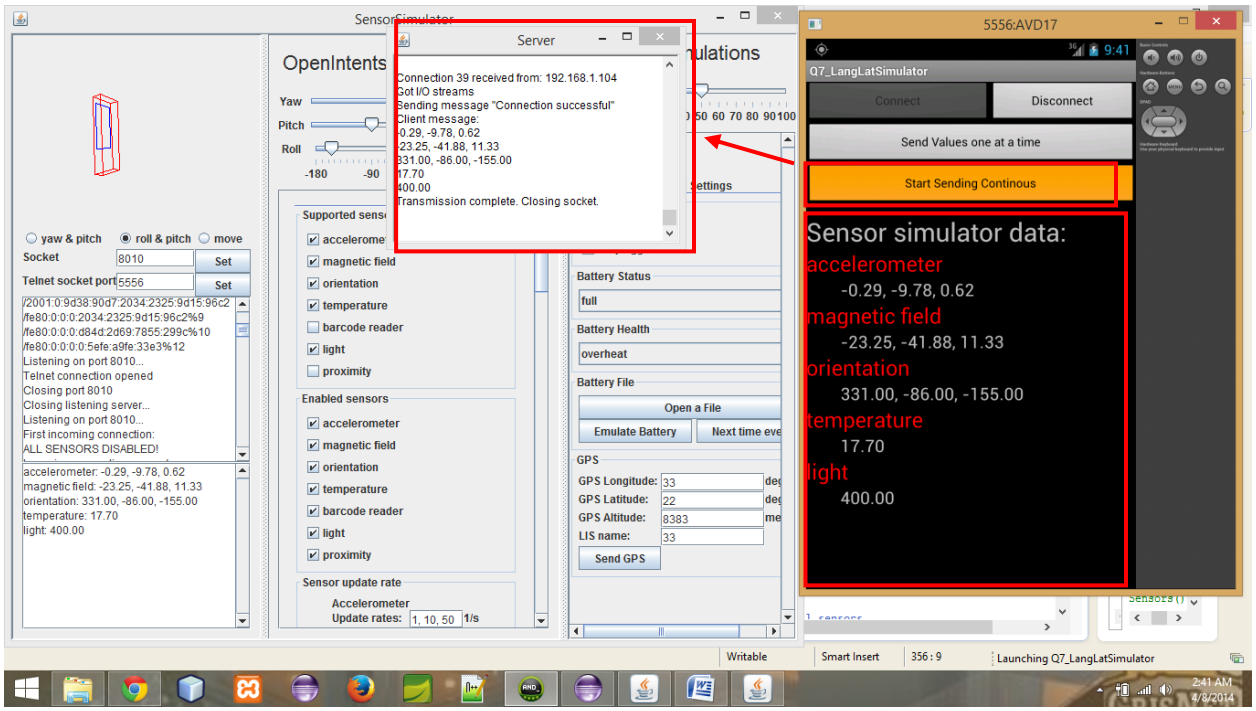



SensorSimulator
OpenIntents Sensor Simulator
Telnet simulations
5556:AVD17
Q7_LangLatSimulator
Connect
Disconnect
Send Values one at a time
Start Sending Continuous
Real device data:
accelerometer
magnetic field
orientation
temperature
barcode reader
light
proximity
Make sure these
Values are same as
Java - Q7_LangLatSimulator/src/org/openintents/sensorsimulator/SensorSimulatorSettingsActivity.java - Eclipse
SensorSimulatorSettingsActivity.java
Server.java
HelloAndroid.java
main.xml
public void connect() {
Log.i(TAG, "Connect");
String newIP = "10.0.2.2"; //mEditTextIP.getText().toString();
String newSocket ="8010"; // mEditTextSocket.getText().toString();
String oldIP = mSensorSimulatorConvenience.getPreference(SensorSimulator.KEY_IPADDRESS);
String oldSocket = mSensorSimulatorConvenience.getPreference(SensorSimulator.KEY_SOCKET);
if (! (newIP.contentEquals(oldIP) && newSocket.contentEquals(oldSocket))) {
// new values
mSensorManager.unregisterListener(listener);
mSensorManager.disconnectSimulator();
// Save the values
mSensorSimulatorConvenience.setPreference(SensorSimulator.KEY_IPADDRESS, newIP);
mSensorSimulatorConvenience.setPreference(SensorSimulator.KEY_SOCKET, newSocket);
}
if (! mSensorManager.isConnectedSimulator()) {
Log.i(TAG, "Not connected yet -> Connect");
mSensorManager.connectSimulator();
}
readAllSensors();
setButtonState();
if (mSensorManager.isConnectedSimulator()) {
mTextSensorType.setText(R.string.sensor_simulator_data);
} else {
mTextSensorType.setText(R.string.real_device_data);
}
Writable
Smart Insert
167 : 65
Launching Q7_LangLatSimulator

Screenshots

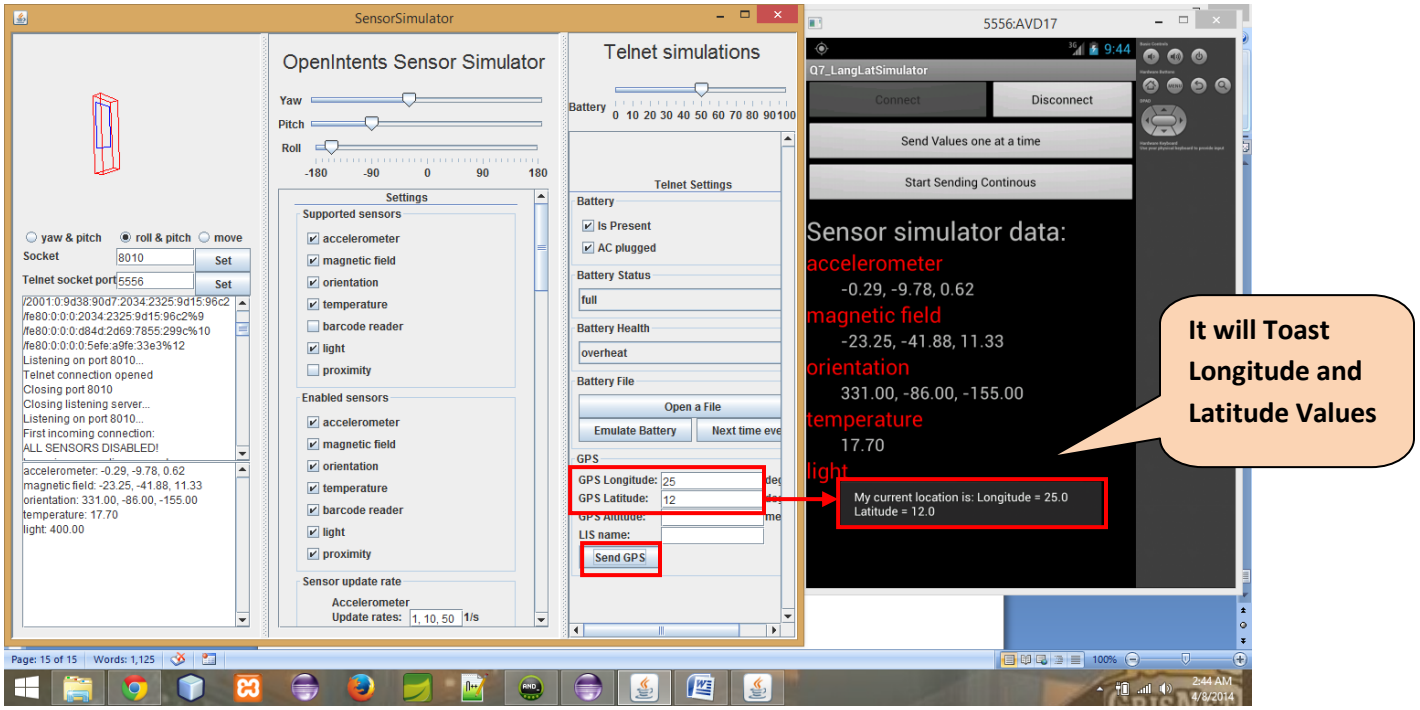






By Cliking Start Sending Contours will keep on sending data to Server.

ou can click Stop Sending Continues to stop sending continuous data to server.



Progrogram is 364 lines code so not attachned here Please find it in Attachment.