**Simulation of Weather Data processing**

**Through Smartphone Sensors**

*A report submitted in partial fulfilment of requirements of project for*

*Bachelor of Technology*

In

Computer Science and Engineering

By

**BATCH-02**

**CLOUD COMPUTING**

**K. ANUSHA M.SAI AJAY**

**(12003030) (12007259)**

**N. LIKITHA N.V.HARIKA**

**(12003596) (12003127)**

Under the Guidance of

**Dr. B. Thirumala Rao**

****

Department of Computer Science and Engineering

K L University, Vaddeswaram

2015-2016

**1. ABSTRACT**

The usage of internet has increased from 52.8% in 2012 to about 83% in 2015. With that the number of smart phone users is increasing drastically as it provides more feature than a mobile phone. Nowadays people finish most of their works through a single touch on their smart phones. The advanced feature of the smart phones is that they are equipped with different kinds of sensors increases the capabilities of computer to interact with the environment. Therefore we can find most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions.

This project involves creation of a mobile application that uses the smart phone sensors as a reference to collect the data. There are many existing applications that provide details about the weather, which collect the weather conditions data either from the weather stations or from the meteorological satellites where the stations are not present. The Android sensor framework which provides several methods to interact with the built-in sensors. The API also provides methods that let you determine the capabilities of each sensor, such as its maximum range, its resolution, and its power requirements. To sense the weather we can use different sensors more specifically barometer sensor which can be used for floor positioning. Previously indoor positioning methods focused on plane locations on a floor and did not provide accurate floor positioning. The values from sensors can’t be directly read from a sensor to a mobile phone but rather we need a mobile application which does acts as an interface between the sensor and the user.

The extension to the application development will be storing these values in the cloud cluster so that they can be processed because the mobile local database will not be sufficient to do so. Once these values are available in the cluster we can process this data using Map Reduce techniques and predict the values in the future to make end-decisions.

**2. PARTIAL COMPLETED WORK FOR PHASE-III**

**2.1 Sensors Overview**

We have to develop a mobile application that will take the data from the built-in smart phone sensors instead of taking the dataset already available. To read the data from the application the sensor related API should be used which can identify the sensors and their capabilities and also can monitor the sensor events occurring on the device on which the app is running. The sensor raw data can be read every time a sensor detects a change in the measuring parameter. It provides the information like name of the sensor, time stamp, accuracy and the value of the sensor data. There are different formats of reading the sensor data; it depends on the format the device uses.

There are many different kinds of sensors like motion sensors, environmental sensors, and position sensors.

* Motion Sensors:

These sensors are used in android platform to facilitate our various requirements for example a motion sensor is used to measure the forces like acceleration, rotational in all the three directions. Accelerometer sensors come under this category. It gives the values in m/s^2 in all the three directions considering the Cartesian co-ordinate system where all the three axes are perpendicular to each other. Apple iphone 3GS is the first to introduce a accelerometer sensors in its phone. The main disadvantage with this sensor is it is difficult to do aircraft applications and for moving platforms.

A gyroscope is generally a physical device consisting of a rotor, gimbals, spinning axis, gyroscope frame which uses the gravitational force of the earth to determine the orientation.

* Environmental sensors

Sensors which are helpful in tracing the environmental aspects are called environmental sensors.

The types include barometer, photometer, and illuminator. Temperature is measured in degrees Celsius, atmospheric pressure in hPa millibars, relative ambient air humidity as a percentage value, and ambient light in SI lux units.

A barometer is a pressure sensor. Sometimes the temperature estimated by satellites is just 36 degrees centigrade but we feel temperature to be 42 degree centigrade that is because of humidity. The ambient temperature parameter has all the features to provide us the temperature since temperature sensor will not be working all the times. It has lot of delay times.

These sensors will be present in Samsung Galaxy s4, s5, s6 and all the latest costly phones we tested on Sony z5 premium. In order to forecast the weather we need to know the pressure and temperature, our requirements will be a phone with embedded pressure and temperature sensor.

* Position sensors

These sensors measure the physical position of a device. Example the output will be like the proximity is 5m. The reference will be taken according to the sensor designer.

Raw data can be collected using android sensor framework. The term “raw data” refers to the data that is collected directly from different sources used during the process. As per our requirements the source considered is sensors i.e. the built in sensors present in the smart phones. This data is also called as “primary data”. The data is just collected but not subjected to any kind of processing or manipulations in this stage the major task involved is collecting of data from the sensors as sensors are the source we are considering. In this project the data from the sensors will be collected through the mobile application installed on the smart phone. The sensor framework provides several classes and interfaces that help you perform a wide variety of sensor-related tasks. The following actions can be performed by a sensor framework:

* Determine which sensors are available on a device. An app which shows the list of sensors present in a cell phone. The following figures are the snapshots of the app taken in Sony Z5 Premium.

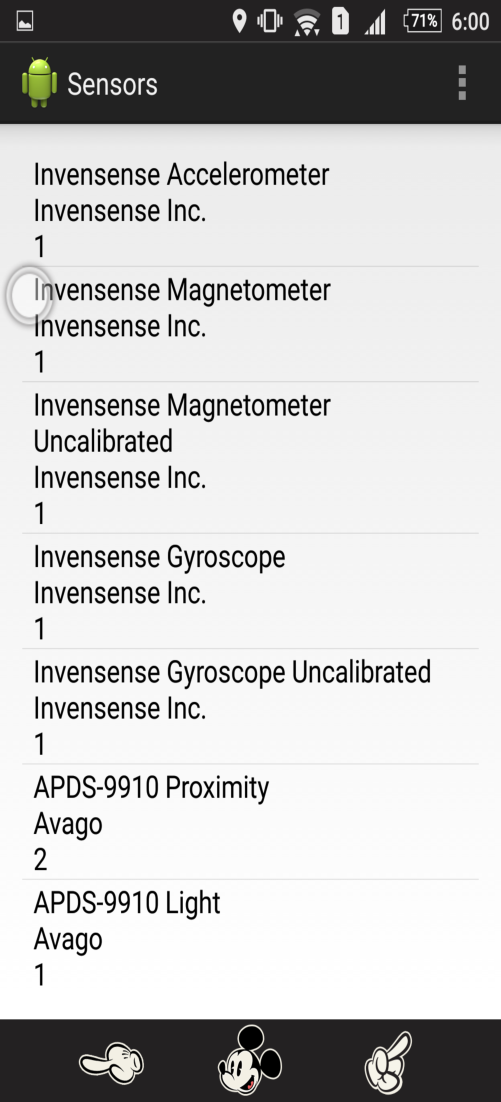


Figure 1

* The above thing also displays individual sensor's capabilities like its manufacturer, power requirements, and Sensor id.
* To obtain raw data and we have functions to minimum time we can get the data for streaming sensors using some methods provided by the pre- defined classes.
* Register and unregister sensor event listeners that monitor sensor changes.

This topic provides an overview of the sensors that are available on the Android platform. It also provides an introduction to the sensor framework.

**2.2 Sensor Framework**

[SensorManager](http://developer.android.com/reference/android/hardware/SensorManager.html)

1. A class to provide instance of a sensor service. To obtain systems service
2. A class used for registering and unregistering sensor event listeners.
3. To tell the sensor accuracy, the time taken to get the data from the sensor.
4. The sensor does not get turned off when the phone screen is off, so we should always disable it since our battery will be wasted if we do not do so because it consumes more power.

[Sensor](http://developer.android.com/reference/android/hardware/Sensor.html)

1. A class for specific type of sensor used to access sensors capabilities.
2. Uses getSensorlist() to list all available sensors of the particular variety.

Parameters: Int – Number of sensors requested.

Return value: a list of sensors matching the asked type. (<list>sensor). The **android.hardware.Sensor** class provides methods to get information of the sensor such as sensor name, sensor type, sensor resolution, sensor type etc.

|  |
| --- |
|  |

[SensorEvent](http://developer.android.com/reference/android/hardware/SensorEvent.html)

1. For providing information like raw sensor data, the type of the sensor, accuracy of the data timestamp for the event about a sensor.
2. This is an instance created by the system.
3. A sensor event occurs every time a sensor detects a change in the parameters it is measuring.

[SensorEventListener](http://developer.android.com/reference/android/hardware/SensorEventListener.html)

1. Interface of SensorEvent class.
2. Two call-back methods as follows:

|  |  |
| --- | --- |
| Methods | Function |
| **void onAccuracyChanged (Sensor sensor, int accuracy)** | It is called when sensor accuracy is changed. |
| **void onSensorChanged (SensorEvent event)** | |  |  | | --- | --- | |  | It is called when sensor values are changed. | |

In a typical application you use these sensor-related APIs to perform two basic tasks:

* **Identifying sensors and sensor capabilities**

Not all phones have the required sensor we needed, our application targeting a particular sensor will not run if the sensor is not present in that phone. These sensors are provided in a phone’s hardware, so we cannot do anything about it. Though there are so many apps in Google play store that give us the value of the weather, they may not necessarily the pressure and temperature sensor, they track our location using GPS and give the weather status using satellites but in our app we make use of environmental sensors to give weather report.

* **Monitor sensor events**

Monitoring refers to data acquisition. We get it when then sensor changes its type time to time, during that we will need to collect the values or outputs we wanted. We will not be using the device temperature sensor, as it is now deprecated as of Android 4.0.Some parameters as mentioned are depreciated or outdated. So it is not preferred to use them. The newer versions are Android studio 1.5 which has many sensors compared to older versions like Android 4.0 since these sensors have been added to studio with time progression. These sensors field in android is still in the budding phase and more sensors will be added in the upcoming versions too.

* 1. **Identifying Sensors and Sensor Capabilities**

The android works using several sensors and different apps use different censors. So it is important to determine the kind and types of censors that present in the mobile as per the mobile company as well as the android version. Because the the censors present in one company are not present in another. For eg: Motorolo Xoom has pressure senso, but the samsung Nexus S does not.It is also important to determine capabilities of each sensor, its maximum range, its resolution and power requirements. We have methods to get a list a every sensor (getSensorList()), find out whether a certain sensor exits on the mobile or not (getDefaultSensor()). the determinants of a sensor are found out by methods like getResolution, getPower etc.

We have two methods so that we can utilize the application to the best irrespective of the mobile manufacture and Android version. One of them is getMinDelay by which we can determine the maximum rate at which a sensor can acquire data. We can manipulate the speed as per the app requirement.

The raw data of the sensors are monitored by invoking the methods onAccuracyChanged() and onSensorChanged(). These provide us with reference to the Sensor object that chnged and the new accuracy of the sensor. The timegap at which the change in data should be observed is called the delay. Care should be taken resume the monitoring when you donot want to monitor lest it should drain the battery.

**2.4 Handling Different Sensor Configurations**

**Detecting sensors at runtime**



Fig 2 : Code to determine the existence of a sensor.

Two ways to check the sensors:

1. Perform checks n whether the user has particular sensors and to avoid using functionality that is totally reliant on them.
2. Google play filters.

**2.5 Best Practices for Accessing and Using Sensors**

1. Unregister sensor listeners:

As mentioned earlier a sensor continues to acquire data even when its activity is paused, so we have to unregister it, to save battery , the below is a code to unregister using the onPause function:

privateSensorManagermSensorManager;  
  ...  
@Override  
protectedvoidonPause(){  
  super.onPause();  
  mSensorManager.unregisterListener (this);  
}

2. Don’t test your code on the emulator

An emulator does not display the outputs; we have to perform it on a physical device. In our project outputs changes from place to place. There is something called sensor simulator which serves the need of testing.

3. Don't block the onSensorChanged() method

Sensor data can change at a high rate, which means the system may call the [onSensorChanged(SensorEvent)](http://developer.android.com/reference/android/hardware/SensorEventListener.html#onSensorChanged(android.hardware.SensorEvent))method quite often. As a best practice,do not stuff too much code in that method since it will delay you, if any code wanted is to be written outside.

4. Avoid using deprecated methods or sensor types

5. Verify sensors before you use them

6. Choose sensor delays carefully

When you register a sensor with the [registerListener()](http://developer.android.com/reference/android/hardware/SensorManager.html#registerListener(android.hardware.SensorEventListener, android.hardware.Sensor, int)) method, be sure you choose a delivery rate that is suitable for your application or use-case. Sensors can provide data at very high rates. Allowing the system to send extra data that you don't need wastes system resources and uses battery power.

**2.6 Install Android Studio**

**System requirements:**

**Windows**

* Microsoft Windows 7/8/10 (32- or 64-bit)
* 2 GB RAM minimum, 8 GB RAM recommended
* 2 GB of available disk space minimum,  
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
* 1280 x 800 minimum screen resolution
* Java Development Kit (JDK) 8
* For accelerated emulator: 64-bit operating system and Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality

**Mac**

* Mac OS X 10.8.5 or higher, up to 10.11.4 (El Capitan)
* 2 GB RAM minimum, 8 GB RAM recommended
* 2 GB of available disk space minimum,  
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
* 1280 x 800 minimum screen resolution
* Java Development Kit (JDK) 6

**Linux**

* GNOME or KDE desktop
* 64-bit distribution capable of running 32-bit applications
* GNU C Library (glibc) 2.11 or later
* 2 GB RAM minimum, 8 GB RAM recommended
* 2 GB of available disk space minimum,  
  4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image)
* 1280 x 800 minimum screen resolution
* Java Development Kit (JDK) 8
* For accelerated emulator: Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality, or AMD processor with support for AMD Virtualization (AMD-V)

**Features:**

1. It can do the “run” operation very fast.
2. The main advantage with android is that it has the intelligence to predict the code and used for code completion, so one need not have to remember all the stuff. When you type a bit of sentence all relevant syntax will syntax will be scrolling down. This feature of android is called intelligent code editor system.
3. In order to test our apps, android provides us with an “emulator” which act like a virtual physical device/phone, Android tablets, Android Wear, and Android TV devices. The newer versions of android have the capabilities to resize the emulator to our convince.
4. Robust and flexible build system
5. There is something called “gradle” that is used to construct our android app system. You have all the source files in the gradle bundle, without gradle bundle you cannot run the application/project.

**Installing Android Studio:**

Setting up Android Studio takes just a few clicks.

While the [Android Studio download](http://developer.android.com/sdk/index.html) completes, verify which version of the JDK you have: open a command line and type javac -version. If the JDK is not available or the version is lower than 1.8, download the [Java SE Development Kit 8](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html).

To install Android Studio on Windows, proceed as follows:

1. Launch the .exe file you downloaded.
2. Follow the setup wizard to install Android Studio and any necessary SDK tools.

On some Windows systems, the launcher script does not find where the JDK is installed. If you encounter this problem, you need to set an environment variable indicating the correct location.

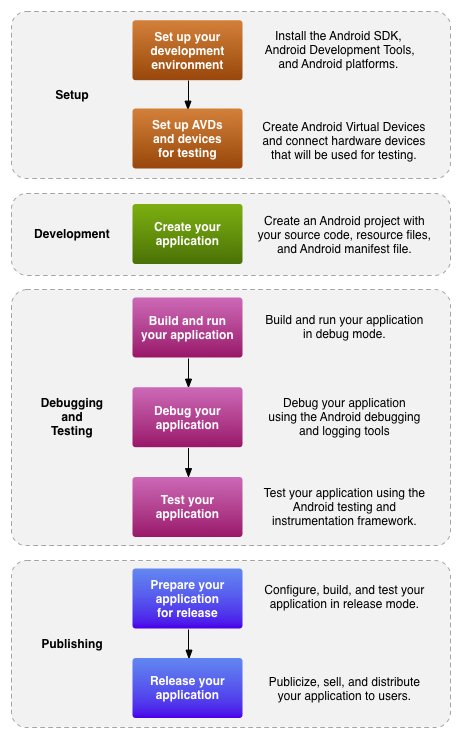
Select **Start menu > Computer > System Properties > Advanced System Properties**. Then open **Advanced tab > Environment Variables** and add a new system variable JAVA\_HOME that points to your JDK folder, for exampleC:\Program Files\Java\jdk1.8.0\_77.

As new tools and other APIs become available, Android Studio tells you with a pop-up, or you can check for updates by clicking **Help > Check for Update**.

Steps to be followed to create your first android application:

* [Creating an Android Project](http://developer.android.com/training/basics/firstapp/creating-project.html)
* [Running Your Application](http://developer.android.com/training/basics/firstapp/running-app.html)
* [Building a Simple User Interface](http://developer.android.com/training/basics/firstapp/building-ui.html)
* [Starting Another Activity](http://developer.android.com/training/basics/firstapp/starting-activity.html)
* In the android studio, we have a folder called ‘res’ which mean resources following folders exists in resources: drawable (contains ic-launcher) in dpi like drawable-hdpi, drawable-mdpi, drawable-xhdp, Drawable-xxhdpi.Before running our project we have to change our project from android to project that is sensors.

# 2.7 App Developer Workflow



**Figure 1.** The development process for Android applications.

To create applications for Android, you utilize an arrangement of apparatuses that are incorporated into Android Studio. Not with standing utilizing the instruments from AndroidIt can straightforwardly conjure the instruments that you require while creating applications.

Be that as it may, you may decide to create with another IDE or a basic content manager and conjure the instruments on the order line or with scripts. This is a less streamlined approach to create in light of the fact that you will once in a while need to call order line devices physically, yet you will have entry to the same number of components that you would have in Android Studio.

## App Workflow

The above figure shows how we develop an app basically, it may or may not be using android studio. In the development phase the following things need to be encompassed:

* **Environment Setup**

This phase includes all the installation process and creating a platform where in you can run your app. You also plug in the physical where you want to download the apk file run it and then view our created android application. We have to click on the SDK manger icon which is visible on the menu bar and we have to download our required SDK packages. In our project we downloaded the SDK API 19 as the minimum version since API 19 and API 21 is more users friendly. The reason why we did not set API minimum level as 8 or something is that it would not be suitable for all applications. API minimum level one to five rarely exists if we set them as minimum we cannot run our application on all platforms so it is not recommendable to use. API 23 is marshmallow, though it is latest, it is not that user friendly so we keep it away. We launch our android application by clicking on the android button.

If the virtual device we want does not exist in the existing list of virtual devices then we have an option to create one of our own choices. We can create a virtual device with API of our choice, the dimension of the virtual device i.e., sometimes bigger virtual devices look clumsy on the screen. So we can choose the company of the device of our requirements.

This phase also includes like setting up a platform which is comfortable for us, we cann hide menus for our convenience coding.

Here the point of download our virtual device comes in phase, we have chosen suitable virtual device depending on our requirements. The device we chose is “NEXUS ONE API 21.”

* **Project Setup and Development**

**We created a new android studio project called “AndroidSensor” with company name “com.android” so the package com.android.androidsensor always appears on the top of every coding xml and java page. We have to do some changes to the resource files before we begin the logic and coding of our project. We specified the project location to be C drive before creating our project.**

**We created a blank activity initially and later on extended that activity by creating two new java classes called pressure activity and temp activity for our convenience and avoidance of crowding of too much code in a single file and defaualt one to be main activity.** During this phase you set up application modules, which contain all of the source code and resource files for your application

**Building, Debugging and Testing**

During this phase we transform our project into a “debuggable .apk” package, her we named “sensor.apk”, apk is a file used for running our app on a physical device so that you can install and run on the emulator or an Android-powered device, it can be phone, tablet, TV anything we chose. Android Studio uses a build system based on [Gradle](http://www.gradle.org/) that provides flexibility, customized build variants, dependency resolution, and much more. If you're using another IDE, you can build your project using Gradle and install it on a device using [adb](http://developer.android.com/tools/help/adb.html).

To maintain a log of the activities we can create a system log with a “logcat”. A layout is applied by the default [Activity](http://developer.android.com/reference/android/app/Activity.html) class that the SDK tools generated when you created the project. In Android Studio, from the toolbar, click **Run**. Or from a command line, change directories to the root of your Android project and execute the following

*$ ant debug*

*adb install -r app/build/outputs/apk/app-debug.apk*

Next, with Android Studio you debug your application using the [AndroidDevice Monitor](http://developer.android.com/tools/help/monitor.html) and device log messages (log cat).

**3. CODE IMPLEMENTATION**

**Mainactivity.java**

package com.android.sensor;

import android.app.Activity;

import android.app.Service;

import android.content.Intent;

import android.hardware.Sensor;

import android.hardware.SensorManager;

import android.os.Bundle;

import android.view.Menu;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ListView;

import android.widget.SimpleAdapter;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

public class MainActivity extends Activity {

private SensorManager sensorManager;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Get the reference to the sensor manager

sensorManager = (SensorManager) getSystemService(Service.SENSOR\_SERVICE);

// Get the list of sensor

//List<Sensor> sensorList = sensorManager.getSensorList(Sensor.TYPE\_ALL);

List<Sensor> sensorList = sensorManager.getSensorList(Sensor.TYPE\_PRESSURE);

List<Map<String, String>> sensorData = new ArrayList<Map<String,String>>();

for (Sensor sensor: sensorList) {

Map<String, String> data = new HashMap<String, String>();

data.put("name", sensor.getName());

data.put("vendor", sensor.getVendor());

sensorData.add(data);

}

SimpleAdapter sa = new SimpleAdapter(this, sensorData, android.R.layout.simple\_list\_item\_2, new String[]{"name", "vendor"}, new int[]{android.R.id.text1, android.R.id.text2});

ListView lv = (ListView) findViewById(R.id.sensorList);

lv.setAdapter(sa);

lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view, int pos,

long id) {

Intent i = new Intent(MainActivity.this, PressActivity.class);

startActivity(i);

}

});

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

}

**Pressactivity.java**

package com.android.sensor;

import android.app.Activity;

import android.app.Service;

import android.hardware.Sensor;

import android.hardware.SensorEvent;

import android.hardware.SensorEventListener;

import android.hardware.SensorManager;

import android.os.Bundle;

import android.widget.TextView;

public class PressActivity extends Activity implements SensorEventListener {

private TextView pressView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(com.android.sensor.R.layout.press\_layout);

pressView = (TextView) findViewById(com.android.sensor.R.id.pressTxt);

// Look for pressure sensor

SensorManager snsMgr = (SensorManager) getSystemService(Service.SENSOR\_SERVICE);

Sensor pS = snsMgr.getDefaultSensor(Sensor.TYPE\_PRESSURE);

snsMgr.registerListener(this, pS, SensorManager.SENSOR\_DELAY\_UI);

}

@Override

protected void onResume() {

super.onResume();

SensorManager snsMgr = (SensorManager) getSystemService(Service.SENSOR\_SERVICE);

Sensor pS = snsMgr.getDefaultSensor(Sensor.TYPE\_PRESSURE);

snsMgr.registerListener(this, pS, SensorManager.SENSOR\_DELAY\_UI);

}

@Override

protected void onStart() {

// TODO Auto-generated method stub

super.onStart();

}

@Override

public void onAccuracyChanged(Sensor sensor, int accuracy) {

}

@Override

public void onSensorChanged(SensorEvent event) {

float[] values = event.values;

pressView.setText("" + values[0]);

}

}

**Tempactivity.java**

package com.android.sensor;

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.widget.TextView;

public class TempActivity extends Activity implements SensorEventListener {

private SensorManager mgr;

private Sensor temp;

private TextView text;

private StringBuilder msg = new StringBuilder(2048);

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(com.android.sensor.R.layout.temp\_layout);

mgr = (SensorManager) this.getSystemService(SENSOR\_SERVICE);

temp = mgr.getDefaultSensor(Sensor.TYPE\_TEMPERATURE);

text = (TextView) findViewById(com.android.sensor.R.id.text);

}

@Override

protected void onResume() {

mgr.registerListener(this, temp, SensorManager.SENSOR\_DELAY\_NORMAL);

super.onResume();

}

@Override

protected void onPause() {

mgr.unregisterListener(this, temp);

super.onPause();

}

public void onAccuracyChanged(Sensor sensor, int accuracy) {

}

public void onSensorChanged(SensorEvent event) {

float fahrenheit = event.values[0] \* 9 / 5 + 32;

msg.insert(0, "Got a sensor event: " + event.values[0] + " Celsius (" +

fahrenheit + " F)\n");

text.setText(msg);

text.invalidate();

}

}

**Activity\_main.xml:**

<?xml version="1.0"?>

[<RelativeLayout tools:context=".MainActivity" android:paddingTop="@dimen/activity\_vertical\_margin" android:paddingRight="@dimen/activity\_horizontal\_margin" android:paddingLeft="@dimen/activity\_horizontal\_margin" android:paddingBottom="@dimen/activity\_vertical\_margin" android:layout\_height="match\_parent" android:layout\_width="match\_parent" xmlns:tools="http://schemas.android.com/tools" xmlns:android="http://schemas.android.com/apk/res/android">](file:///C:\Users\vamsi\Desktop\Sensor\app\src\main\res\layout\activity_main.xml)<ListView android:layout\_height="match\_parent" android:layout\_width="match\_parent" android:id="@+id/sensorList"/></RelativeLayout>

**Press\_layout.xml:**

<?xml version="1.0" encoding="UTF-8"?>

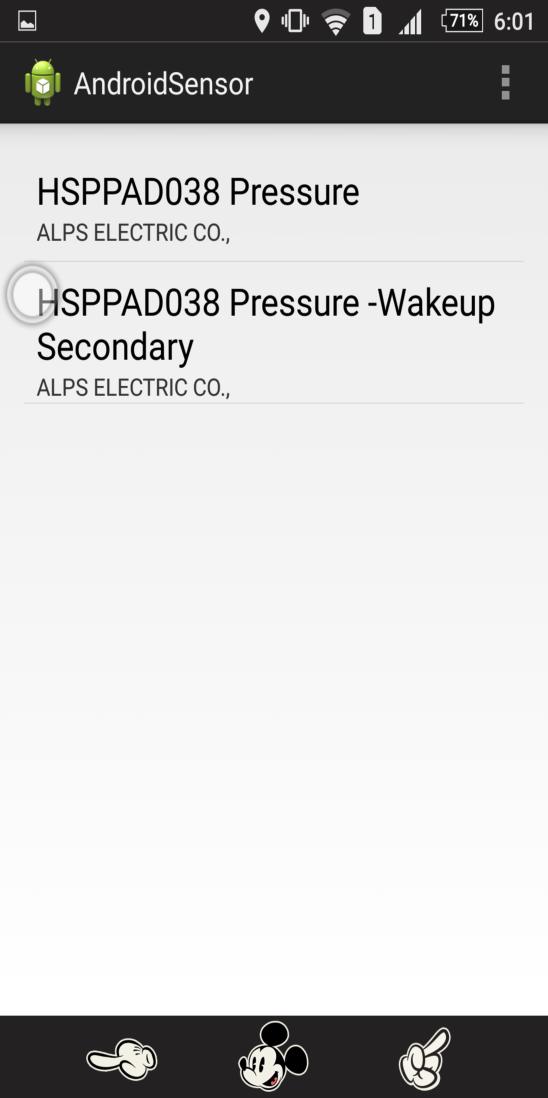
[<RelativeLayout android:layout\_height="match\_parent" android:layout\_width="match\_parent" xmlns:android="http://schemas.android.com/apk/res/android">](file:///C:\Users\vamsi\Desktop\Sensor\app\src\main\res\layout\press_layout.xml)<TextView android:layout\_height="wrap\_content" android:layout\_width="wrap\_content" android:layout\_centerHorizontal="true" style="?android:attr/textAppearanceLarge" android:text="Pressure (mbar)" android:id="@+id/text"/>[<LinearLayout android:layout\_height="80dp" android:layout\_width="160dp" android:id="@+id/l1" android:background="#d3e4f4" android:layout\_centerInParent="true">](file:///C:\Users\vamsi\Desktop\Sensor\app\src\main\res\layout\press_layout.xml)<TextView android:layout\_height="wrap\_content" android:layout\_width="wrap\_content" style="@style/pressStyle" android:id="@+id/pressTxt" android:gravity="center\_horizontal"/></LinearLayout></RelativeLayout>

**Temp\_layout.xml:**

<?xml version="1.0" encoding="UTF-8"?>

[<RelativeLayout android:layout\_height="match\_parent" android:layout\_width="match\_parent" xmlns:android="http://schemas.android.com/apk/res/android">](file:///C:\Users\vamsi\Desktop\Sensor\app\src\main\res\layout\temp_layout.xml)<TextView android:layout\_height="wrap\_content" android:layout\_width="wrap\_content" android:layout\_centerHorizontal="true" style="?android:attr/textAppearanceLarge" android:text="Temperature" android:id="@+id/text"/>[<LinearLayout android:layout\_height="80dp" android:layout\_width="160dp" android:id="@+id/l1" android:background="#d3e4f4" android:layout\_centerInParent="true">](file:///C:\Users\vamsi\Desktop\Sensor\app\src\main\res\layout\temp_layout.xml)<TextView android:layout\_height="wrap\_content" android:layout\_width="wrap\_content" style="@style/tempStyle" android:id="@+id/pressTxt" android:gravity="center\_horizontal"/></LinearLayout></RelativeLayout>

**SCREENSHOTS**

  
 Fig 3: Output showing the Pressure sensor Menu

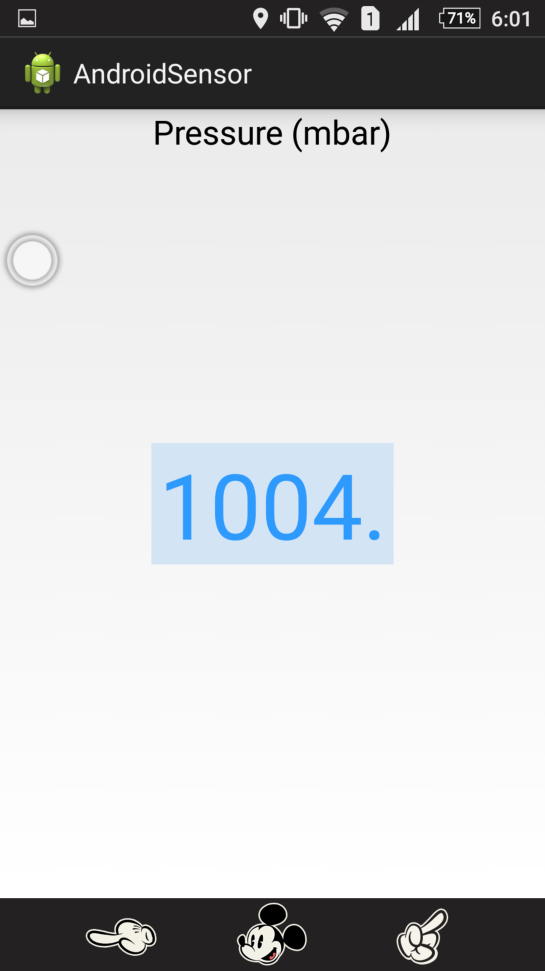


Fig 4: Output showing the value of the

pressure sensors In mbar.