USF UNIVERSITY OF SOUTH FLORIDA
COLLEGE OF ENGINEERING

Using the Android Sensor API

Juan José Marrón
Department of Computer Science & Engineering
jmarronm@mail.usf.edu

Outline

- Sensors description:
  - Motion Sensors
  - Environmental Sensors
  - Positioning Sensors
- Android Sensor API
- SensorTest Application
- Implementation

Sensors Description

- A **sensor** is a converter that measures a physical quantity and converts it into a signal which can be read by an observer (person) or by instrument (electronic device).

  Ex: Thermometer

- Most Android devices have been manufactured with sensors that measure motion, orientation, and even environmental conditions.

  Ex: Accelerometers are utilized by the platform to switch from screen vertical to horizontal orientation

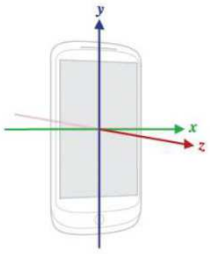
- Some of these sensors are hardware-based and some are software-based.

- Few Android-powered devices have every type of sensor.

Motion Sensors

- These sensors measure acceleration forces and rotational forces along three axes. Useful for monitoring device movement, such as tilt, shake, rotation, or swing.
- This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.

- An **acceleration** sensor measures the acceleration in m/s2 applied to the device, including the force of gravity
- The **gyroscope** measures the rate or rotation in rad/s around a device's x, y, and z



Environmental Sensors

- These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.
- Environment sensors are not always available on devices.

- The **illumination** sensor detects the illuminance level and measures the result in lx units

Position sensors

- These sensors measure the physical position of a device. They are useful for determining a device's physical position in the world's frame of reference.
- This category includes orientation sensors and magnetometers.

- The **orientation** sensor lets you monitor the position of a device relative to the earth's frame of reference. (Azimuth, Pitch and Roll)
- The **geomagnetic field** sensor lets you monitor changes in the earth's magnetic field. This sensor provides raw field strength data (in μT) for each of the three coordinate axes.

Android Sensor API

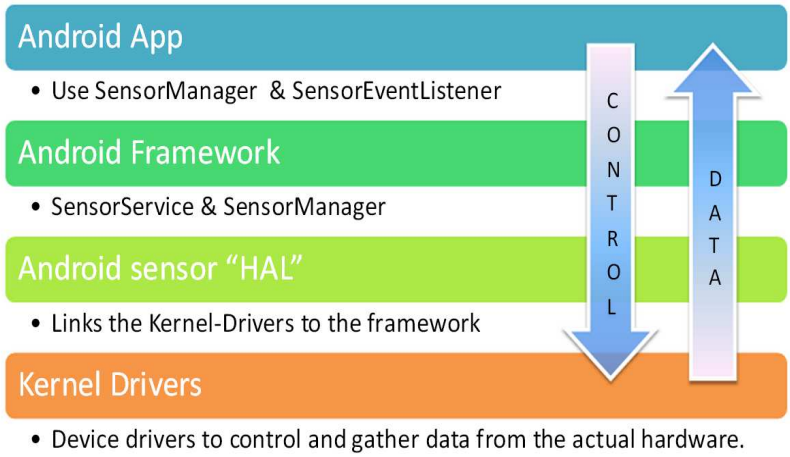
- The Sensor API framework is part of the *android.hardware* package and it hides aspects on how physically are the sensors attached to the device and low other level details.

  Ex: How to handle interruptions or to read the sensors data.

- The Sensor framework provides support for hardware features and it allows developers to:
  - Identify the sensors and their capabilities.
  - Monitor sensor events.
  - Handle different sensor configurations.

- These possibilities are provided through several methods thanks to the implementation of the interfaces:
  - SensorManager
  - Sensor
  - SensorEvent
  - SensorEventListener

Android Sensor API



Android Sensor API

- The functionality of the Sensor API interfaces is:
- SensorManager

  Creates an instance of the sensor service. Provides methods for accessing and listing sensors, registering/unregistering sensor event listeners. Also sensor constants to report sensor accuracy, set data acquisition rates, and calibrate sensors.
- Sensor

  Creates an instance of a specific sensor.

  This class provides methods to determine a sensor's capabilities.
- SensorEvent

  This class creates a sensor event object, which provides information about: the raw sensor data, the type of sensor that generated the event, the accuracy of the data, and the timestamp for the event.
- SensorEventListener

  Creates callback methods that receive notifications when sensor values change or when sensor accuracy changes.

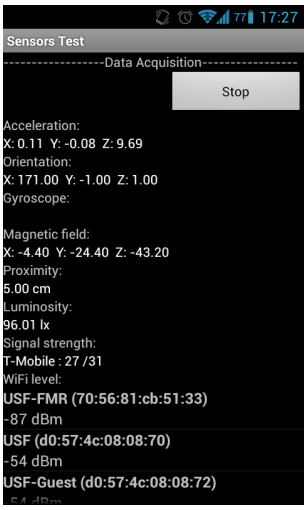
SensorTest Application

- Basic example of using the Android Sensor API.
- SensorTest is a simple unique activity app gathering and showing information of several sensors in a mobile device.
- Motion Sensors:

  Accelerometers, gyroscopes
- Environmental Sensors:

  Light sensor, proximity sensor, WiFi level and Signal strength detectors
- Position Sensors:

  Orientation sensors and magnetometers

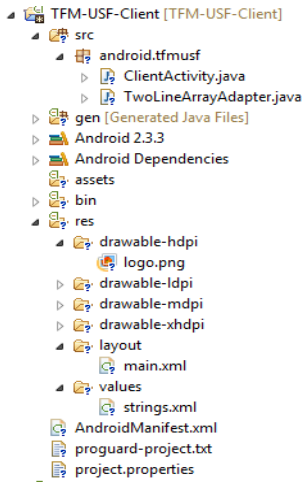


SensorTest Application

Project Structure

- Source (src)
  - ClientActivity.java
- Resources (res)
  - Drawable = logo.png
  - Layout = main.xml
  - Values = strings.xml
- AndroidManifest.xml

ANDROID SENSORS



SensorTest Application

AndroidManifest.xml [code]

- Permissions

```
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

- Application

```
<application
    android:icon="@drawable/Logo"
    android:label="@string/app_name" >
```

- Activity

```
<activity
    android:name=".ClientActivity"
    android:label="@string/app_name" >
```

SensorTest Application

Layout - main.xml [code]

- LinearLayout:

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
```

- TextViews:

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/aceleration" />
```

- Buttons

- ScrollView:

```
<ScrollView
  android:id="@+id/scrollView1"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content" >
```

- ListView:

```
<ListView
  android:id="@+id/wifiListView"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  tools:listitem="@android:Layout/
  simple_list_item_2" >
</ListView>
```


SensorTest Application

Values – Strings.xml [code]

- Resources

```
<resources>
   <string name="main">---------Data Acquisition----------</string>
   <string name="app_name">Sensors Test</string>
   <string name="startButton">Start</string>
   <string name="stopButton">Stop</string>
   <string name="results">Sensor results:</string>
   <string name="acceleration">Acceleration:</string>
   <string name="orientation">Orientation:</string>
   <string name="gyroscope">Gyroscope:</string>
   <string name="magneticfield">Magnetic field:</string>
   <string name="proximity">Proximity:</string>
   <string name="luminosity">Luminosity:</string>
   <string name="signalstrength">Signal strength:</string>
   <string name="wifilevel">WiFi level:</string>
</resources>
```

SensorTest Application

src – ClientActivity.java [code]

- Imports

```
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
```

- Classes

```
public class ClientActivity extends Activity  implements SensorEventListener  {...}

private class SignalStrengthListener extends PhoneStateListener {...}

private class WifiReceiver extends BroadcastReceiver {...}
```


Implementation

- In general to utilize a sensor, the process is as follows:

1. Obtain a reference of the *SensorManager* provided by the app context via *getSystemService()*.
2. Implement the *SensorListener* interface and *registerListener()* to get its notifications in *onResume()* and unregister in *onPause()*.
3. The notifications come via *onSensorChanged()* and *onAccuracyChanged()* having its own unique ID constant to be identified.
4. The sensor values and format we get in *onSensorChanged()* will depend on the specific sensor.

Implementation

- Accelerometer, gyroscope, orientation sensor, magnetometer, proximity and luminosity sensor follow a similar implementation pattern.

- Only one *SensorManager* for handling all of the sensors

```
private SensorManager sensorManager;
```

- Get the SensorManager instance in *onCreate() as SENSOR_SERVICE*

```
public void onCreate(Bundle savedInstanceState) {
   sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);}
```

- Register the listener in *onResume()*

```
protected void onResume() {
    sensorManager.registerListener(this,
          sensorManager.getDefaultSensor(Sensor.TYPE_XXXXX),
          SensorManager.SENSOR_DELAY_GAME);
```

Constants ID: `TYPE_ACCELEROMETER, TYPE_ORIENTATION, TYPE_GYROSCOPE, TYPE_PROXIMITY, TYPE_LIGHT, TYPE_MAGNETIC_FIELD`

Implementation

- Receive values in *onSensorChanged()*

Ex: Accelerometer

```
public void onSensorChanged(SensorEvent event) {
          switch (event.sensor.getType())
          {
             case Sensor.TYPE_ACCELEROMETER:
                float xA = event.values[0];
                float yA = event.values[1];
                float zA = event.values[2];

                String accelerationOut =
                            String.format("X: %.2f  Y: %.2f  Z: %.2f", xA, yA, zA);
                 accelerationView.setText(accelerationOut);

             break;
          }
}
```

Implementation

- Receive values in *onSensorChanged()*

Ex: Proximity

```
public void onSensorChanged(SensorEvent event) {
          switch (event.sensor.getType())
         {
            case Sensor.TYPE_PROXIMITY:
                 float distance = event.values[0];

                 String proximityOut = String.format("%.2f cm", distance);
                 proximityView.setText(proximityOut);

            break;
          }
}
```


Implementation

- Signal Strength treatment

- Imports

```
import android.telephony.PhoneStateListener;
import android.telephony.TelephonyManager;
```

- *TelephonyManager* and *SignalStrengthListener* for handling the reception

```
private TelephonyManager  telephonyManager;
private SignalStrengthListener signalStrengthListener;
```

- Get the *telephonyManager* instance in *onCreate() as TELEPHONY_SERVICE*

```
public void onCreate(Bundle savedInstanceState) {
     telephonyManager = (TelephonyManager)getSystemService(TELEPHONY_SERVICE);
     carrierName = telephonyManager.getNetworkOperatorName(); }
```

Implementation

- Register the listener in *onResume()*

```
protected void onResume() {
    signalStrengthListener = new SignalStrengthListener();
    telephonyManager.listen(
          signalStrengthListener,SignalStrengthListener.LISTEN_SIGNAL_STRENGTHS); }
```

- Receive values in method *onSignalStrengthsChanged()* of *SignalStrengthListener* class

```
public void onSignalStrengthsChanged(android.telephony.SignalStrength signalStrength)
{
    int strengthAmplitude = signalStrength.getGsmSignalStrength();
    String strengthsignalOut = carrierName +
                          String.format(" : %02d /31",strengthAmplitude);
     signalstrengthView.setText(strengthsignalOut);
 }
```


Implementation

- WiFi Level treatment

- Imports

```
import android.net.wifi.ScanResult;
import android.net.wifi.WifiManager;
```

- *WifiManager* and *WifiReceiver* for handling the reception

```
private WifiManager mainWifi;
private WifiReceiver receiverWifi;
private List<ScanResult> wifiList;
```

- Get the instances in *onCreate() as WIFI_SERVICE*

```
public void onCreate(Bundle savedInstanceState) {
     mainWifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);
     receiverWifi = new WifiReceiver();
}
```

Implementation

- Register the listener in *onResume()*

```
protected void onResume() {
        registerReceiver(receiverWifi,
              new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)); }
```

- StartScan method for asking for results

```
mainWifi.startScan();
```

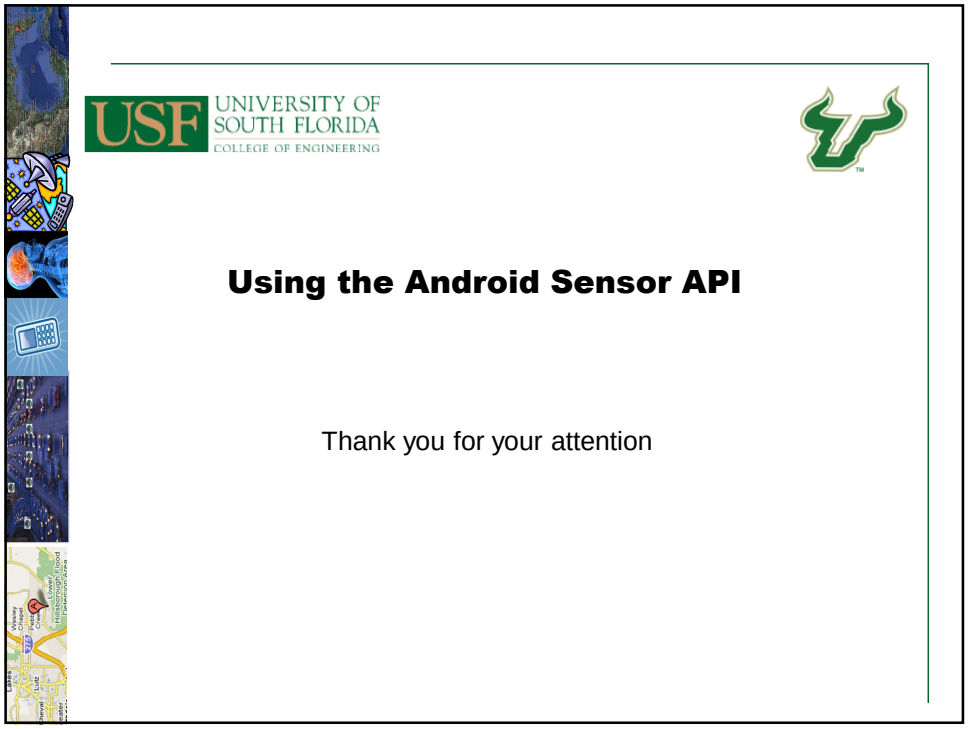
- Receive values in method *onReceive()* of *WifiReceiver* class

```
public void onReceive(Context c, Intent intent) {
       wifiList = mainWifi.getScanResults();
       Wifi[] wifis = new Wifi[wifiList.size()];
       for (int i = 0; i < wifiList.size(); i++){
       String WifiID = wifiList.get(i).SSID + " (" + wifiList.get(i).BSSID + ")";
       wifis[i] = new Wifi( WifiID, wifiList.get(i).level +" dBm"); }
}
```


Implementation

- Unregister listeners and receivers in *onPause()*

```
protected void onPause() {
      sensorManager.unregisterListener
            (this, sensorManager.getDefaultSensor(Sensor.TYPE_XXXXXX));
      unregisterReceiver(receiverWifi);
      signalStrengthListener.unregister();
}
```


USF
UNIVERSITY OF
SOUTH FLORIDA
COLLEGE OF ENGINEERING
Using the Android Sensor API
Thank you for your attention