

Sensors

Using sensors

Contents

<u>CONTENTS</u>	<u>2</u>
<u>LISTING THE SENSORS</u>	<u>3</u>
<u>PROXIMITY SENSOR</u>	<u>3</u>
<u>ACCELEROMETER</u>	<u>5</u>

Listing the sensors

In this example we will learn how to list the sensors available in a mobile device. First of all we have to create a new Project.

Once the project has been created, we have to create a *SensorManager* object, which allows managing the sensors of the device.

```
SensorManager sm = (SensorManager)
this.getSystemService(Context.SENSOR_SERVICE);
```

Then we have to list the different available sensors in the device:

```
Iterator<Sensor> i = sm.getSensorList(Sensor.TYPE_ALL).iterator();
String result = new String();
while (i.hasNext()) {
    Sensor s = i.next();
    result += "\nSensor: " + s.getName();
}
```

In this string called *“result”* all names of the sensors will be available. For showing this information we will use the default TextView (HelloWorld).

We have to indicate an id to the TextView for being able to use it in the activity.

If we test this example in the emulator (the emulator does not have any sensor) the string *“result”* will be empty, so we will write a message with the following code:

```
TextView textView1 = (TextView) findViewById(R.id.textView1);
if (result.isEmpty())
    textView1.setText("Sensors not found ");
else
    textView1.setText(result);
```

Run the project and test the results.

Proximity sensor

Proximity sensor allows knowing if there is any object close to the device (or close to the position of the sensor). This sensor is usually located at the top-front of the device.

This example will change the background of the app when an object is close to the proximity sensor.

First of all we have to create a new project (or use a previous project) and we have to edit the default layout file: *“activity_main.xml”*

```
<?xml version="1.0" encoding="utf-8"?>
<View xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffff">
```

```
</View>
```

We are creating a View object and we will change its background color with the proximity sensor.

Next, in the “*MainActivity*”, we have to create three class attributes (before the “*onCreate*” method).

```
View view;  
SensorManager mSensorManager;  
SensorEventListener mSensorListener;
```

The “*View*” object will be linked to the View of the layout, object “*SensorManager*” will manage the proximity sensor and “*SensorEventListener*” will handle the events of the sensor, by receiving the corresponding values.

The following line of code (in “*onCreate*”) will link the *View* of the layout with the “*View*” object.

```
view = findViewById(R.id.main);
```

We have to get the sensors service with “*SensorManager*” as follows:

```
mSensorManager = (SensorManager)  
getSystemService(Context.SENSOR_SERVICE);
```

Then we initialize “*SensorEventListener*”, which will handle the events of the sensor:

```
mSensorListener = new SensorEventListener() {  
    public void onAccuracyChanged(Sensor sensor, int accuracy) {  
    }  
  
    public void onSensorChanged(SensorEvent se) {  
        float x = se.values[0];  
        Log.d("PROXIMITY", "" + x);  
        if (x == 0) {  
            changeRed();  
        } else {  
            changeWhite();  
        }  
    }  
};
```

When we initialize “*SensorEventListener*” we observe that we have to implement the “*onAccuracyChanged*” and “*onSensorChanged*” methods: the first one is used for informing when the sensor accuracy changes, and the second one will be called when the sensor values change.

With the sentence “*se.values[0]*” we are getting the value of the proximity sensor, i.e., the distance to the object.

Finally, we are calling the “*changeRed()*” method when the object is close to the sensor, or the “*changeWhite()*” method if the object is not close to the sensor.

Once this is done, we have to register the listener in the “*onStart*” method and unregister it in the “*onStop*” method for saving battery when the activity is closed or hidden.

We write the “*onStart()*” method, and we also check if the device has a proximity sensor or not.

```
@Override
protected void onStart() {
    super.onStart();
    Log.i("ProximityTest", "onSTART");
    Sensor
    sensor=mSensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
    if (sensor!=null)
        mSensorManager.registerListener(mSensorListener, sensor,
    SensorManager.SENSOR_DELAY_UI);
}
```

The *onStop()* method is as follows:

```
@Override
protected void onStop() {
    Log.i("ProximityTest ", "onSTOP");
    mSensorManager.unregisterListener(mSensorListener);
    super.onStop();
}
```

Finally we have to implement the “*changeRed*” and “*changeWhite*” methods for changing the View background color.

```
private void changeRed()
{
    view.setBackgroundColor(Color.RED);
}

private void changeWhite()
{
    view.setBackgroundColor(Color.WHITE);
}
```

Now we can test the example in a real device.

Accelerometer

In the following example we will use the accelerometer sensor. For doing this we have to create a new Project and edit the default layout (*activity_main.xml*).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
```

```
android:orientation="vertical">

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

<LinearLayout
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/x" />

    <TextView
        android:id="@+id/x"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>

<LinearLayout
    android:id="@+id/LinearLayout2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/y" />

    <TextView
        android:id="@+id/y"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>

<LinearLayout
    android:id="@+id/LinearLayout3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/z" />
```

```

        <TextView
            android:id="@+id/z"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

    </LinearLayout>
</LinearLayout>

```

We are using different `LinearLayout` and `TextViews` for showing the sensor values on the screen.

We have to create two class attributes:

```

SensorManager mSensorManager;
SensorEventListener mSensorListener;

```

In the “*onCreate*” method we have to write the following line of code:

```

mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

```

Then we have to initialize “*SensorEventListener*”:

```

mSensorListener = new SensorEventListener() {
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    public void onSensorChanged(SensorEvent se) {
        float[] values = se.values;
        TextView x = (TextView) findViewById(R.id.x);
        TextView y = (TextView) findViewById(R.id.y);
        TextView z = (TextView) findViewById(R.id.z);
        x.setText(String.valueOf(values[0]));
        y.setText(String.valueOf(values[1]));
        z.setText(String.valueOf(values[2]));
    }
};

```

In this case, the accelerometer sensor will return three values: X, Y and Z. They will be written in the “*TextViews*” created above.

In the “*onStart()*” method we have to register *SensorEventListener* and check if the sensor is available in the device:

```

@Override
public void onStart()
{
    super.onStart();
    Sensor sensor =
mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    if (sensor!=null)
        mSensorManager.registerListener(mSensorListener, sensor,
SensorManager.SENSOR_DELAY_FASTEST);
}

```

```
}
```

In the “*onStop()*” method we have to unregister the listener:

```
@Override  
public void onStop()  
{  
    mSensorManager.unregisterListener(mSensorListener);  
    super.onStop();  
}
```