# Sensors Overview

* An overview on sensors
* Some method of the class [SensorManager](https://developer.android.com/reference/android/hardware/SensorManager.html) for accessing and listing sensors
* Example of a project about the [Accelerometer](https://developer.android.com/reference/android/hardware/Sensor.html) Sensor
* Example of a project about some Environmental Sensors.

Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions. These sensors are capable of providing raw data with high precision, and accuracy, and are useful if you want to monitor three-dimensional device movement or positioning, or you want to monitor changes in the ambient environment near a device. For example, a game might track readings from a device's gravity sensor to infer complex user gestures and motions, such as tilt, shake, rotation, or swing. Likewise, a weather application might use a device's temperature sensor and humidity sensor to calculate and report the dewpoint, or a travel application might use the geomagnetic field sensor and accelerometer to report a compass bearing.

The Android platform supports **three broad categories of sensors**:

## Motion sensors

These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.

## 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.

The raw data you acquire from the light, pressure, and temperature sensors usually requires no calibration, filtering, or modification, which makes them some of the easiest sensors to use.

## Position sensors

These sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.

You can access sensors available on the device and acquire raw sensor data by using the Android sensor framework. The sensor framework provides several classes and interfaces that help you perform a wide variety of sensor-related tasks.

The Android sensor framework lets you access many types of sensors. Some of these sensors are hardware-based and some are software-based. Hardware-based sensors are physical components built into a handset or tablet device. They derive their data by directly measuring specific environmental properties, such as acceleration, geomagnetic field strength, or angular change. Software-based sensors are not physical devices, although they mimic hardware-based sensors. Software-based sensors derive their data from one or more of the hardware-based sensors and are sometimes called virtual sensors or synthetic sensors. The linear acceleration sensor and the gravity sensor are examples of software-based sensors.

Few Android-powered devices have every type of sensor. For example, most handset devices and tablets have an accelerometer and a magnetometer, but fewer devices have barometers or thermometers. Also, a device can have more than one sensor of a given type. For example, a device can have two gravity sensors, each one having a different range.

Sensor types supported by the Android platform are [TYPE\_ACCELEROMETER](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_ACCELEROMETER), [TYPE\_AMBIENT\_TEMPERATURE](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_AMBIENT_TEMPERATURE), [TYPE\_GRAVITY](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_GRAVITY), [TYPE\_GYROSCOPE](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_GYROSCOPE), [TYPE\_LIGHT](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_LIGHT), [TYPE\_LINEAR\_ACCELERATION](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_LINEAR_ACCELERATION), [TYPE\_MAGNETIC\_FIELD](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_MAGNETIC_FIELD), [TYPE\_ORIENTATION](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_ORIENTATION), [TYPE\_PRESSURE](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_PRESSURE), [TYPE\_PROXIMITY](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_PROXIMITY), [TYPE\_RELATIVE\_HUMIDITY](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_RELATIVE_HUMIDITY), [TYPE\_ROTATION\_VECTOR](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_ROTATION_VECTOR), [TYPE\_TEMPERATURE](https://developer.android.com/reference/android/hardware/Sensor.html#TYPE_TEMPERATURE) (see their description in[[1]](#footnote-1) )

## Sensor Framework

You can access these sensors and acquire raw sensor data by using the Android sensor framework. The sensor framework is part of the [android.hardware](https://developer.android.com/reference/android/hardware/package-summary.html) package and includes the some classes and interfaces. The class, SensorManager, provides various methods for accessing and listing sensors, registering and unregistering sensor event listeners, and acquiring orientation information. This class also provides several sensor constants that are used to report sensor accuracy, set data acquisition rates, and calibrate sensors.

Determine which sensors are available on a device.

## The First Project: The list of the Sensors

To acquire data from the sensors you first create an instance of the SensorManager class, which you can use to get an instance of a physical sensor. Then you register a sensor listener in onResume() method, and start handling incoming sensor data in the onSensorChanged() callback method.

We can develop a simple example that gives us the list of sensors that we have in our device

Let create a new project in Android Studio, name it SensorList and choose Basic Activity option. In layout we use a list control.

<**ListView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:id="@+id/listview1"**/>

We manage the list control from MainActivity as follows

**package ….**

**import** android.hardware.Sensor;  
**import** android.hardware.SensorManager;  
**import** android.os.Bundle;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.support.v7.widget.Toolbar;  
**import** android.util.Log;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.ListView;  
**import** android.widget.Toast;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 SensorManager **mgr**;  
 ListView **listview**;  
 List<String> **liststring** ;  
 List<Sensor> **sensors**;  
 ArrayAdapter<String> **adapter** ;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 Toolbar toolbar = (Toolbar) findViewById(R.id.***toolbar***);  
 setSupportActionBar(toolbar);  
  
  
 **mgr** = (SensorManager) getSystemService(***SENSOR\_SERVICE***);  
 **sensors** = **mgr**.getSensorList(Sensor.***TYPE\_ALL***);  
 displaySsensorList();  
 }  
  
 **void** displaySsensorList(){  
 **listview**=(ListView) findViewById(R.id.***listview1***);  
  
 **liststring** = **new** ArrayList<String>();

//liststring Array is populated with sensor names as values  
 **for** (Sensor sensor : **sensors**) {  
 Log.*d*(**"Sensors"**, **""** + sensor.getName());  
 **liststring**.add(sensor.getName());  
  
 }  
  
 **adapter** = **new** ArrayAdapter<String>(MainActivity.**this**,  
 android.R.layout.***simple\_list\_item\_2***,  
 android.R.id.***text1***, **liststring** );  
 **listview**.setAdapter(**adapter**);  
 }

. . .

}

## 

If we want to select one of the list items we need to add a list listener

**import** android.hardware.Sensor;  
**import** android.hardware.SensorManager;  
**import** android.os.Bundle;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.support.v7.widget.Toolbar;  
**import** android.util.Log;  
**import** android.view.Menu;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.ListView;  
**import** android.widget.Toast;  
  
**import** java.util.ArrayList;  
**import** java.util.List;

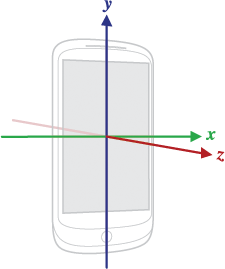
**public class** MainActivity **extends** AppCompatActivity {  
 . . .   
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **. . .**

displaySsensorList();  
 **listview**.setOnItemClickListener(**new** AdapterView.OnItemClickListener()  
 {  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View view,  
 **int** position, **long** id) {  
 Toast.*makeText*(MainActivity.**this**, **liststring**.get(position),

Toast.***LENGTH\_SHORT***).show();  
 }  
 });  
 }  
. . .

}

## [Accelerometer](https://developer.android.com/reference/android/hardware/Sensor.html) Sensor

In general, the sensor framework uses a standard 3-axis coordinate system to express data values. Accelerometer sensor is one of them. For most sensors, the coordinate system is defined relative to the device's screen when the device is held in its default orientation.

When a device is held in its default orientation, the X axis is horizontal and points to the right, the Y axis is vertical and points up, and the Z axis points toward the outside of the screen face. In this system, coordinates behind the screen have negative Z values.

The Android accelerometer data specification is: Sensor.TYPE\_ACCELEROMETER. All values are in units (m/s2) and measure the acceleration to the phone minus the force of gravity.

* values[0]: Acceleration minus Gx on the x-axis
* values[1]: Acceleration minus Gy on the y-axis
* values[2]: Acceleration minus Gz on the z-axis

For example, when the device lies flat on a table and is pushed on its left side toward the right, the x acceleration value is positive. When the device lies flat on a table, the acceleration value is +9.81, which correspond to the acceleration of the device (0 m/s2) minus the force of gravity (-9.81 m/s2).

|  |  |  |  |
| --- | --- | --- | --- |
| Position | X | Y | Z |
| UP: | 0 | 9.81m/sec2 | 0 |
| LEFT: | 9.81m/sec2 | 0 | 0 |
| DOWN: | 0 | -9.81m/sec2 | 0 |
| RIGHT: | -9.81m/sec2 | 0 | 0 |
| FRONT UP: | 0 | 0 | +9.81m/sec2 |
| BACK UP: | 0 | 0 | -9.81m/sec2 |

You can access the sensor via the sensorManager.getDefaultSensor() method, which takes the sensor type and the delay defined as constants on SensorManager, as parameters.

### Sensor Listener

Once you acquired a sensor, you can register a SensorEventListener object on it. This listener will get informed, if the sensor data changes.

To avoid the unnecessary usage of battery power, you can register your listener in the onResume() method and de-register it in the onPause() method.

### Accelerometer Example

We will build an application which will change its background color if it is shuffled. Create a new Android project called *SensorTestActivity* with an *activity* called *MainActivity*.

Change your layout file to the following code.

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"** >  
  
 <**TextView  
 android:id="@+id/textView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:text="Shake to get a toast and to switch color"** />  
  
</**LinearLayout**>

Change your activity class to the following code.

**package** . . .  
  
**import** android.app.Activity;  
**import** android.graphics.Color;  
**import** android.hardware.Sensor;  
**import** android.hardware.SensorEvent;  
**import** android.hardware.SensorEventListener;  
**import** android.hardware.SensorManager;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.view.Window;  
**import** android.view.WindowManager;  
**import** android.widget.Toast;  
  
**public class** MainActivity **extends** Activity **implements** SensorEventListener {  
 **private** SensorManager **sensorManager**;  
 **private boolean color** = **false**;  
 **private** View **view**;  
  
 */\*\* Called when the activity is first created. \*/* @Override  
 **public void** onCreate(Bundle savedInstanceState) {  
 requestWindowFeature(Window.***FEATURE\_NO\_TITLE***);  
 getWindow().setFlags(WindowManager.LayoutParams.***FLAG\_FULLSCREEN***,  
 WindowManager.LayoutParams.***FLAG\_FULLSCREEN***);  
  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **view** = findViewById(R.id.***textView***);  
 **view**.setBackgroundColor(Color.***GREEN***);  
  
 **sensorManager** = (SensorManager) getSystemService(***SENSOR\_SERVICE***);  
  
 }  
  
 @Override  
 **public void** onSensorChanged(SensorEvent event) {  
 **if** (event.**sensor**.getType() == Sensor.***TYPE\_ACCELEROMETER***) {  
 getAccelerometer(event);  
 }  
  
 }  
  
 **private void** getAccelerometer(SensorEvent event) {  
 **float**[] values = event.**values**;  
 *// Movement* **float** x = values[0];  
 **float** y = values[1];  
 **float** z = values[2];  
  
 **float** accelationSquareRoot = (x \* x + y \* y + z \* z)  
 / (SensorManager.***GRAVITY\_EARTH*** \*SensorManager.***GRAVITY\_EARTH***);  
  
 **if** (accelationSquareRoot >= 2) *//* {  
 Toast.*makeText*(**this**, **"Device was shuffed"**, Toast.***LENGTH\_SHORT***)  
 .show();  
 **if** (**color**) {  
 **view**.setBackgroundColor(Color.***GREEN***);  
 } **else** {  
 **view**.setBackgroundColor(Color.***RED***);  
 }  
 **color** = !**color**;  
 }  
 }  
  
 @Override  
 **public void** onAccuracyChanged(Sensor sensor, **int** accuracy) {  
  
 }  
  
 @Override  
 **protected void** onResume() {  
 **super**.onResume();  
 *// register this class as a listener for the orientation and  
 // accelerometer sensors* **sensorManager**.registerListener(**this**,  
 **sensorManager**.getDefaultSensor(Sensor.***TYPE\_ACCELEROMETER***),  
 SensorManager.***SENSOR\_DELAY\_NORMAL***);  
 }  
  
 @Override  
 **protected void** onPause() {  
 *// unregister listener* **super**.onPause();  
 **sensorManager**.unregisterListener(**this**);  
 }  
}

Notes:

It is often required software full-screen display, custom title (using buttons and other controls) and other needs. Set the full screen:

// No title

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

//full screen

getWindow().setFlags(WindowManager.LayoutParams. FLAG\_FULLSCREEN , WindowManager.LayoutParams. FLAG\_FULLSCREEN);

### An example about Environmental sensors, Light, Pressure, and Temperature Sensors

The Android platform provides four sensors that let you monitor various environmental properties. You can use these sensors to monitor relative ambient humidity, illuminance, ambient pressure, and ambient temperature near an Android-powered device.

Unlike most motion sensors and position sensors, which return a multi-dimensional array of sensor values for each SensorEvent, environment sensors return a single sensor value for each data event.

The raw data you acquire from the light, pressure, and temperature sensors usually requires no calibration, filtering, or modification, which makes them some of the easiest sensors to use. To acquire data from these sensors you first create an instance of the SensorManager class, which you can use to get an instance of a physical sensor. Then you register a sensor listener in the onResume() method, and start handling incoming sensor data in the onSensorChanged() callback method. The following code shows how to do this:

Let organize the work of sensors, each of them in one intent which is called from MainActivity. Here we have three buttons each of them will activate one of intents. The layout of the MainActivity is as follows:

*<?***xml version="1.0" encoding="utf-8"***?>*<**RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/activity\_main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingBottom="@dimen/activity\_vertical\_margin"  
 android:paddingLeft="@dimen/activity\_horizontal\_margin"  
 android:paddingRight="@dimen/activity\_horizontal\_margin"  
 android:paddingTop="@dimen/activity\_vertical\_margin"  
 tools:context="com.example.user.sensorenvironment.MainActivity"**>  
  
 <**Button  
 android:id="@+id/Light"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="@string/light"  
 android:onClick="LightData"**/>  
  
 <**Button  
 android:id="@+id/Pressure"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/Light"  
 android:text="@string/pressure"  
 android:onClick="PressureData"**/>  
  
 <**Button  
 android:id="@+id/Temperature"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/Pressure"  
 android:text="@string/temperature"  
 android:onClick="TemperatureData"**/>  
  
 <**TextView  
 android:id="@+id/textview"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/Temperature"  
 android:text=""** />  
  
</**RelativeLayout**>

We see that each button has the attribute **android:onClick=". . .",** which substitute in code the use of the listener for the button.

Beside the buttons we have here a text field where we will display the result that each intent will give to us.

For each button we will have a method in MainActivity code that will activate the corresponding intent, so we will have the following code for this class.

package . . .

**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.TextView;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 **static final int *LIGHT\_REQUEST*** = 1;  
 **static final int *PRESSURE\_REQUEST*** = 2;  
 **static final int *TEMPERATURE\_REQUEST*** = 3;  
 Button **btn1**,**btn2**,**btn3**;  
 TextView **textView**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 **textView**=(TextView)findViewById(R.id.***textview***);  
 **btn1**= (Button)findViewById(R.id.***Light***);  
 **btn2**= (Button)findViewById(R.id.***Pressure***);  
 **btn1**= (Button)findViewById(R.id.***Temperature***);  
 }  
  
 **void** LightData(View view) {  
 Intent lightIntent=**new**

Intent(MainActivity.**this**,LightMeasurement.**class**);  
 startActivityForResult(lightIntent,***LIGHT\_REQUEST*** );  
 startActivity(lightIntent);  
 }  
 **void** PressureData(View view) {  
 Intent pressureIntent=**new**

Intent(MainActivity.**this**,PressureMeasurement.**class**);

startActivityForResult(pressureIntent,***PRESSURE\_REQUEST*** );  
 startActivity(pressureIntent);  
 }  
 **void** TemperatureData(View view) {  
 Intent temperatureIntent=**new**

Intent(MainActivity.**this**,TemperatureMeasurement.**class**);  
 startActivityForResult(temperatureIntent,***TEMPERATURE\_REQUEST*** );  
 startActivity(temperatureIntent);  
 }  
 @Override  
 **protected void** onActivityResult(**int** requestCode, **int** resultCode,

Intent data) {  
 *// Check which request we're responding to* **if** (requestCode == ***LIGHT\_REQUEST***) {  
 *// Make sure the request was successful* **if** (resultCode == ***RESULT\_OK***) {  
 **textView**.

setText(data.getStringExtra(LightMeasurement.***EXTRA\_RESPONSE***));  
}  
 }  
 }  
}

what is new for us is the way how we have activated an intent in each of the button methods. For example, the method that is executed when we click on the Light button, has inside it the code:

Intent lightIntent=**new**

Intent(MainActivity.**this**,LightMeasurement.**class**);  
 startActivityForResult(lightIntent,***LIGHT\_REQUEST*** );  
 startActivity(lightIntent);

startActivityForResult() method is used when we expect the get back some information from the intent. There's nothing special about the Intent object you use when starting an activity for a result, but you do need to pass an additional integer argument to the startActivityForResult() method, in our case the constant ***LIGHT\_REQUEST***.

The integer argument is a "request code" that identifies your request. When you receive the result Intent, the callback provides the same request code so that your app can properly identify the result and determine how to handle it in the method onActivityResult().

**protected void** onActivityResult(**int** requestCode, **int** resultCode,

Intent data) {  
 *// Check which request we're responding to* **if** (requestCode == ***LIGHT\_REQUEST***) {  
 *// Make sure the request was successful* **if** (resultCode == ***RESULT\_OK***) {  
 **textView**.

setText(data.getStringExtra(LightMeasurement.***EXTRA\_RESPONSE***));  
}  
 }  
 }

The parameter requestCode corresponds to the constant ***LIGHT\_REQUEST***, that we are discussing.

The parameter resultCode comes from the Intent where it takes the value ***RESULT\_OK.***

The parameter data is of the type Intent and it is used for the information that we need to get from the Intent.

Let see now the Intent code that manages the Light sensor.

package . . .   
**import** android.app.Activity;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.hardware.Sensor;  
**import** android.hardware.SensorEvent;  
**import** android.hardware.SensorEventListener;  
**import** android.hardware.SensorManager;  
**import** android.os.Bundle;  
**import** android.util.Log;  
  
**public class** LightMeasurement **extends** Activity **implements** SensorEventListener {  
 **static final** String ***EXTRA\_RESPONSE*** =**"result"**;  
 **private** SensorManager **mSensorManager**;  
 **private** Sensor **mLight**;  
 String **result**=**""**;  
 **int nr**=0;  
  
 @Override  
 **public final void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 *// Get an instance of the sensor service,*

*// and use that to get an instance of a particular sensor.* **mSensorManager** = (SensorManager)

getSystemService(Context.***SENSOR\_SERVICE***);  
 **mLight** = **mSensorManager**.getDefaultSensor(Sensor.***TYPE\_LIGHT***);  
 }  
  
 @Override  
 **public final void** onAccuracyChanged(Sensor sensor, **int** accuracy) {  
 *// Do something here if sensor accuracy changes.* }  
  
 @Override  
 **public final void** onSensorChanged(SensorEvent event) {  
 **float** light\_intensity = event.**values**[0];  
 **result** += **"\n"** + light\_intensity;  
  
 **mSensorManager**.unregisterListener(**this**);  
 Intent intent = **new** Intent(**this**, MainActivity.**class**);  
 intent.putExtra(***EXTRA\_RESPONSE***, **result**);  
 setResult(***RESULT\_OK***, intent);  
 finish();

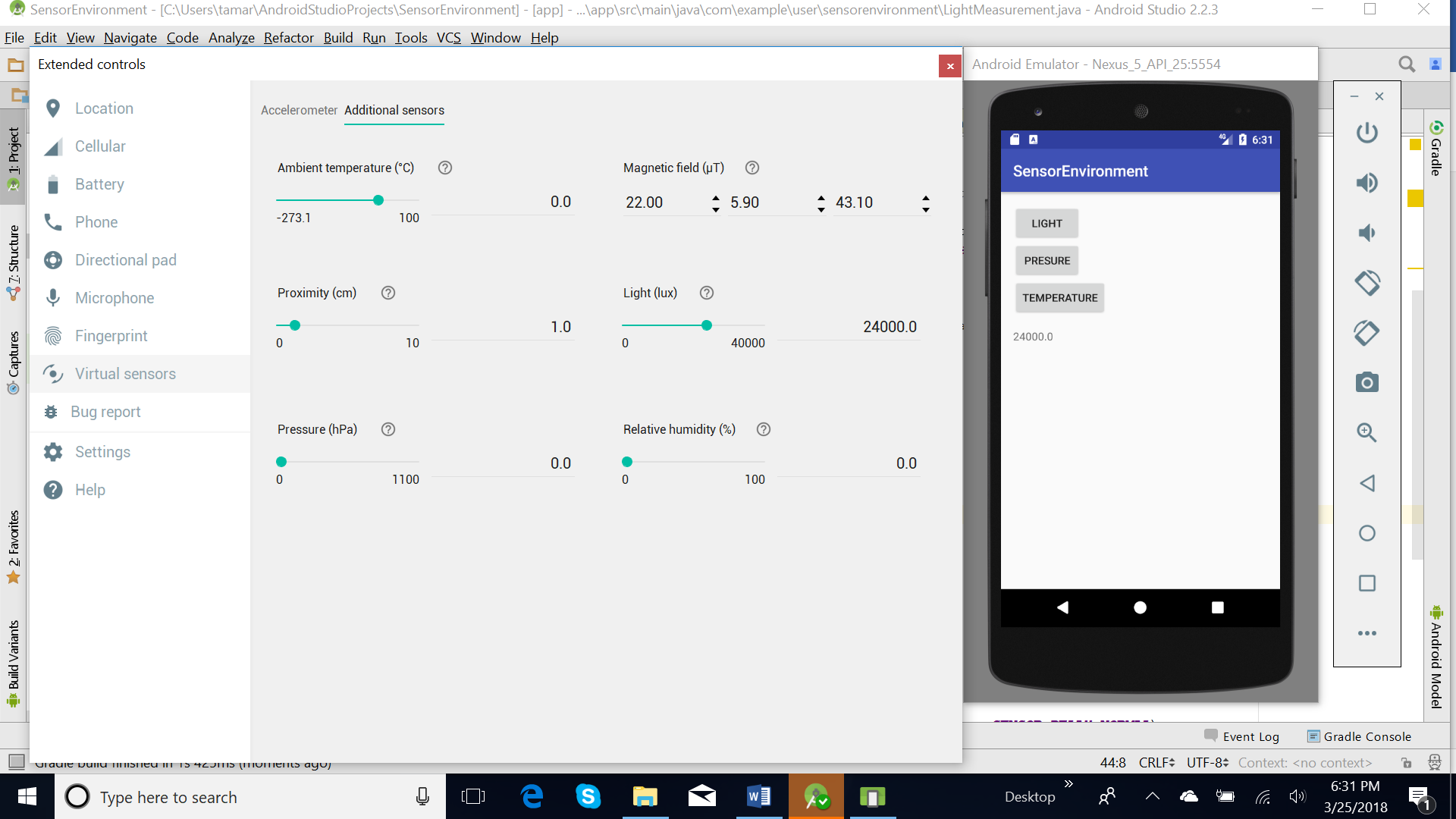
}  
  
 @Override  
 **protected void** onResume() {  
 *// Register a listener for the sensor.* **super**.onResume();  
 **mSensorManager**.registerListener(

**this**, **mLight**, SensorManager.***SENSOR\_DELAY\_NORMAL***);  
 }  
  
 @Override  
 **protected void** onPause() {  
 **super**.onPause();  
 *// Be sure to unregister the sensor when the activity*

*// pauses.super.onPause();* **mSensorManager**.unregisterListener(**this**);  
 }  
}

You must always include implementations of both the onAccuracyChanged() and onSensorChanged() callback methods. Also, be sure that you always unregister a sensor when an activity pauses. This prevents a sensor from continually sensing data and draining the battery.

The method is activated every time the sensor catches a change in the light intensity.



In similar way you can complete the code for the other intents, which correspond to Pressure and Temperature sensors management.

### Assignment

Combine the above examples (first and third one). Use the list of sensors. When we click on of the list item we call an intent where the specific sensor data are displayed.

Use just Main Activity and another Activity similar with the LightMeasurement that we used in the last example.

We call an intent, we associate it with liststring.get(position)and this string is analyzed in the second activity. Depending of this value, different sensors can be considered.

### References

<https://developer.android.com/guide/topics/sensors/sensors_overview.html>

http://www.vogella.com/tutorials/AndroidSensor/article.html

<http://cache.freescale.com/files/sensors/doc/app_note/AN4317.pdf>

https://androidkennel.org/passing-data-between-activities-using-intents-android-tutorial/

1. https://developer.android.com/guide/topics/sensors/sensors\_overview.html [↑](#footnote-ref-1)