**Week 11: Sensor in Android**

The first app lists the available sensors on the device or emulator. The list of sensors is scrollable, if it is too big to fit the screen

## Part 1: List the available sensors

1. Create a new Android project. Call it **SensorSurvey** and use the Empty Views activity template.
2. Open res/layout/activity\_main.xml.
3. Add a margin of 16 dp to the constraint layout.
4. Delete the existing TextView.
5. Add a [ScrollView](https://developer.android.com/reference/android/widget/ScrollView.html" \t "_blank) element inside the constraint layout. Give it these attributes.

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| android:layout\_width | "match\_parent" |
| android:layout\_height | "match\_parent" |
| app:layout\_constraintBottom\_toBottomOf | "parent" |
| app:layout\_constraintTop\_toTopOf | "parent" |
| app:layout\_constraintLeft\_toLeftOf | "parent" |
| app:layout\_constraintRight\_toRightOf | "parent" |

1. Add a TextView element inside the ScrollView and give it these attributes:

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| android:id | "@+id/sensor\_list" |
| android:layout\_width | "wrap\_content" |
| android:layout\_height | "wrap\_content" |
| android:text | "(placeholder)" |

1. Open MainActivity and add a variable at the top of the class to hold an instance of SensorManager:

private var mSensorManager: SensorManager? = null

1. In the onCreate() method, below the setContentView() method, get an instance of the sensor manager from system services, and assign it to the mSensorManager variable:

*//get the sensor manager* mSensorManager = getSystemService(*SENSOR\_SERVICE*) as SensorManager

1. Get the list of all sensors from the sensor manager. Store the list in a [List](https://developer.android.com/reference/java/util/List.html) object whose values are of type [Sensor](https://developer.android.com/reference/android/hardware/Sensor.html):

val sensorList = mSensorManager!!.getSensorList(Sensor.*TYPE\_ALL*)

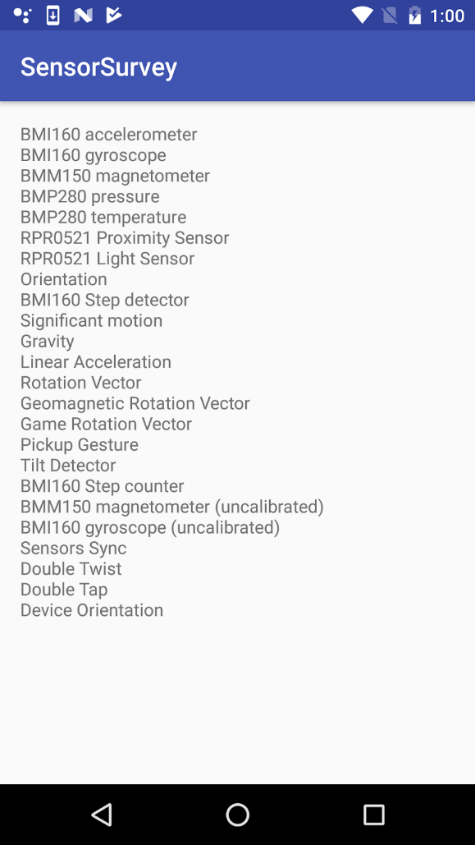
1. Iterate over the list of sensors. For each sensor, get that sensor's official name with the [getName()](https://developer.android.com/reference/android/hardware/Sensor.html" \l "getName()" \t "_blank) method, and append that name to the sensorText string. Each line of the sensor list is separated by the value of the line.separator property, typically a newline character:

val sensorText = StringBuilder()  
  
for (currentSensor in sensorList) {  
 sensorText.append(currentSensor.*name*).append(  
 System.getProperty("line.separator")  
 )  
}

1. Get a reference to the TextView for the sensor list, and update the text of that view with the string containing the list of sensors.

val sensorTextView = findViewById<View>(R.id.sensor\_list) as TextView  
sensorTextView.*text* = sensorText

1. Run the app on a physical device. The output of the app looks something like this screenshot.



## Part 2: Get Sensor Data

1. Create a new empty views activity project called **SensorListener**.
2. Open res/layout/activity\_main.xml.

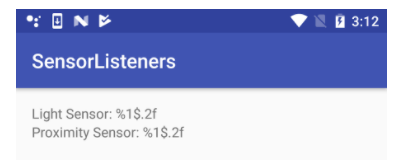
1. Delete the default TextView element.
2. Add a TextView and give it the attributes in the following table. Extract the string into a resource called "label\_light". This text view will print the current value from the light sensor.

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| android:id | "@+id/label\_light" |
| android:layout\_width | "wrap\_content" |
| android:layout\_height | "wrap\_content" |
| android:text | "Light Sensor: %1$.2f" |
| app:layout\_constraintLeft\_toLeftOf | "parent" |
| app:layout\_constraintTop\_toBottomOf | "parent" |

1. Copy and paste the TextView element. Change the attributes in the following table. Extract the string into a resource called "label\_proximity". This text view will print values from the proximity sensor.

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| android:id | "@+id/label\_proximity" |
| android:text | "Proximity Sensor: %1$.2f" |
| app:layout\_constraintTop\_toBottomOf | "@+id/label\_light" |

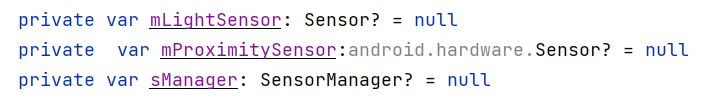
1. The layout for your app should look like this screenshot:



1. Open res/values/strings.xml and add this line:

<string name="light\_text">Light Sensor: %1$.2f</string>  
<string name="proximity\_text">Proximity Sensor: %1$.2f</string>  
<string name="error\_no\_sensor">No sensor</string>

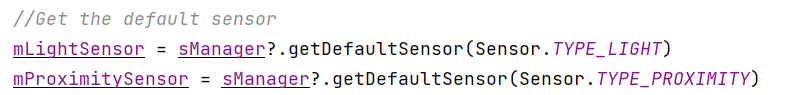
1. Configure the binding object in the MainActivity  class.
2. Open MainActivity and add private member variables at the top of the class to hold [Sensor](https://developer.android.com/reference/android/hardware/Sensor.html) objects for the light and proximity sensors.



1. In the onCreate() method, initialise the sensor manager object right after the setContentView statement.

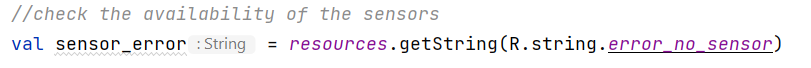


1. Get instances of the default light and proximity sensors. These will be instances of the Sensor class. Assign them to their respective variables:



The [getDefaultSensor()](https://developer.android.com/reference/android/hardware/SensorManager.html" \l "getDefaultSensor(int)" \t "_blank) method is used to query the sensor manager for sensors of a given type. The sensor types are defined by the Sensor class. If there is no sensor available for the given type, the getDefaultSensor() method returns null

1. Get the error string you defined earlier from the strings.xml resource:



1. Test that there is an available light sensor. If the sensor is not available (that is, if getDefaultSensor() returns null), set the display text for the light sensor's TextView to the error string.

if (mLightSensor == null) {  
 binding!!.textLight.*text* = sensor\_error  
}

1. Test for the existence of the proximity sensor.

if (mProximitySensor == null) {  
 binding!!.textProximity.*text* = sensor\_error  
}

1. At the top of the class, modify the class signature to implement the SensorEventListener interface.

class MainActivity : AppCompatActivity(), **SensorEventListener** {

1. Click the red light bulb icon, select "implement methods," and select all methods.
2. Use the onStart() and onStop() methods to register and unregister your sensor listeners.



A single call to the SensorManager. [unregisterListener()](https://developer.android.com/reference/android/hardware/SensorManager.html" \l "unregisterListener(android.hardware.SensorEventListener)" \t "_blank) method unregisters all the registered listeners.

1. In the [onSensorChanged()](https://developer.android.com/reference/android/hardware/SensorEventListener.html" \l "onSensorChanged(android.hardware.SensorEvent)" \t "_blank) method, get the sensor type:

val sensorType: Int = event!!.sensor.getType()

1. Also in onSensorChanged(), get the sensor value.

val currentValue: Float = event!!.values.get(0)

1. Add a when statement for the sensorType variable. Add a case for Sensor.TYPE\_LIGHT to indicate that the event was triggered by the light sensor.

when (sensorType) {  
 Sensor.*TYPE\_LIGHT* -> binding!!.textLight.*text* =  
 getString(R.string.*light\_text*, currentValue)  
  
 else -> {}  
}

1. Add a second case for the proximity sensor ( [Sensor.TYPE\_PROXIMITY](https://developer.android.com/reference/android/hardware/Sensor.html" \l "TYPE_PROXIMITY" \t "_blank)).

Sensor.*TYPE\_PROXIMITY* -> {  
  
 if (currentValue <= 4 && currentValue >=-4)  
 binding!!.textProximity.*text* = "Near"  
 else  
 binding!!.textProximity.*text* = "Far"  
}

1. Run the app on a physical device.