CS 473 - MDP
Mobile Device Programming

Maharishi International University

CS 473 - MDP
Mobile Device Programming

MS.CS Program

Department of Computer Science

Renuka Mohanraj , Ph.D.

Maharishi International
University

CS 473 – MDP
Mobile Device Programming

Lesson 11
Sensors

Maharishi International
University

Wholeness of the lesson

This lesson introduces you to the wide range of sensors that can be built into an Android device which measure motion, orientation, and various environmental conditions. Gives the fundamental knowledge to work with sensor and sensor framework. *The most fundamental knowledge is the most important knowledge since everything else is built upon it. The reason TM can provide such a wide range of benefits to life in general is because it works at such a truly fundamental level.*

Agenda

- Introduction
- Sensor Type
- Sensor Framework
- Sensor Implementation Structure
- Hands on Example 1 – Retrieve Device Sensor List
- Hands on Example 2 - TYPE_ACCELEROMETER
- Hands on Example 3 & 4 - TYPE_LIGHT

Android and Sensors

- Sensors can discover user action and respond
- Device contents rotate as needed
- Walking adjusts position on map
- Tilting steers, a virtual car or controls a physical toy
- Moving too fast disables game interactions

Introduction

- Most of the Android powered devices come with the default sensors such as:
- **Motion Sensors:** These sensors measure acceleration forces along the three axis. This includes accelerometers, gravity sensors, rotation vector, and gyroscopes(movements).
  - An example of an app that uses these types of sensors is a fitness tracker app that keeps track of the steps taken throughout the day or your general activity level.
- **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.
  - An example of an app that uses these sensors may be a hiking app that displays temperature, and air pressure information.
- **Position Sensors**: These sensors measure the physical position of the device which includes orientation changes and magnetometers.

Sensor types supported by the Android platform

Sensor	Description	Common Uses
TYPE_ACCELEROMETER	Measures the acceleration force in m/s2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.	Motion detection (shake, tilt, etc.).
TYPE_LIGHT	Measures the ambient light level (illumination) in lx.	Controlling screen brightness.
TYPE_ORIENTATION	Measures degrees of rotation that a device makes around all three physical axes (x, y, z).	Determining device position.

Refer for more sensors: https://developer.android.com/guide/topics/sensors/sensors_overview.html

Sensor Framework

- You can access sensors and acquire raw sensor data by using the Android sensor framework. The sensor framework is part of the android.hardware package and includes the following classes and interfaces:
- **SensorManager**
  - The SensorManager class handles the usage of sensors and can be invoked by the method, Context.getSystemService().
  - provides various methods for accessing and listing sensors, registering and unregistering sensor event listeners, and acquiring orientation information.
  - provides several sensor constants that are used to report sensor accuracy, set data acquisition rates, and calibrate sensors.

Sensor Framework

- **Sensor**
  - You can use this class to create an instance of a specific sensor. This class provides various methods that let you determine a sensor's capabilities.
- **SensorEvent**
  - This class stores information about the sensor type, sensor data, and values measured by the Sensor etc.,
- **SensorEventListener**
  - 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 register your listener in the onResume() method and unregister it in the onPause() method.

Sensor Implementation Template

```
class SensorActivity : Activity(), SensorEventListener {
    private lateinit var sensorManager: SensorManager
    private var mLight: Sensor? = null

    public override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main)

        sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
        mLight = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT)
    }

    override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) {
        // Do something here if sensor accuracy changes.
    }

    override fun onSensorChanged(event: SensorEvent) {
        // The light sensor returns a single value.
        // Many sensors return 3 values, one for each axis.
        val lux = event.values[0]
        // Do something with this sensor value.
    }

    override fun onResume() {
        super.onResume()
        mLight?.also { light ->
            sensorManager.registerListener(this, light, SensorManager.SENSOR_DELAY_NORMAL)
        }
    }

    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
}
```

Steps to work with Sensor

1. Implementing the SensorEventListener Interface in your Activity

```
class SensorActivity : AppCompatActivity(), SensorEventListener
```

2. Get a reference to the SensorManager to work on Sersor service

```
var mySensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager;
```

3. To determine if the sensor we are trying to use is available on the device model the app is running on, we need to try to get the sensor reference and check if it is null:

```
var tempSensor =
mySensorManager.getDefaultSensor(Sensor.TYPE_AMBIENT_TEMPERATURE);
if (tempSensor != null) {
  // The sensor exists
} else {
  // The sensor does not exist
}
```

Steps to work with Sensor

4. **Registering a listener to receive sensor data by overriding onResume().**

mySensorManager.registerListener(this, mySensors,
SensorManager.SENSOR_DELAY_NORMAL)

Arguments

arg1 → SensorEventListener listener

arg2→ Sensor sensor

arg3→ int samplingPeriodUs

Values for the arg3 are

SENSOR_DELAY_FASTEST (default int value 0 - get sensor data as fast as possible)

SENSOR_DELAY_GAME (default int value 1 - rate suitable for games)

SENSOR_DELAY_UI (default int value 2-rate suitable for the user interface like rotating the screen orientation)

SENSOR_DELAY_NORMAL (default int value 3 - rate (default) suitable for screen orientation changes)

Steps to work with Sensor

5. **Unregistering the sensor by Overriding onPause()**
   – Needed to save your battery life

```
mySensorManager.unregisterListener(this)
```

Steps to work with Sensor

6. To monitor raw sensor data you need to implement two callback methods that are exposed through the
SensorEventListener interface:
The Android system calls these methods whenever the following occurs:

```
  override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) {
        // Do something here if sensor accuracy changes.
    }
override fun onSensorChanged(event: SensorEvent) {
/*  Do something with this sensor event - object contains information about the sensor data*/
}
```

Steps to work with Sensor

Usage of onAcuuracyChanged() code

```
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy){
switch(accuracy){
    case SensorManager.SENSOR_STATUS_ACCURACY_HIGH:
    this.accuracy.setText("SENSOR_STATUS_ACCURACY_HIGH");
    break;
    case SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM:
    this.accuracy.setText("SENSOR_STATUS_ACCURACY_MEDIUM");
    break;
    case SensorManager.SENSOR_STATUS_ACCURACY_LOW:
    this.accuracy.setText("SENSOR_STATUS_ACCURACY_LOW");
    break;
    case SensorManager.SENSOR_STATUS_UNRELIABLE:
    this.accuracy.setText("SENSOR_STATUS_UNRELIABLE");
    break; }}
```

Hands on Example – 1- Sensor List

- Problem Requirement
  - To get the list of sensors from your device and display in the ListView.
  - Create an xml file with one ListView component and configure id .
  - Write your logic to retrieve the list of available sensors in MainActivity.kt

MainActivity.kt

```
import android.content.Context
import android.hardware.Sensor
import android.hardware.SensorManager
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.ArrayAdapter
import android.widget.ListView
class MainActivity : AppCompatActivity() {
    lateinit var listView: ListView
    lateinit var sensorManager: SensorManager
    lateinit var listsensor:List<Sensor>
    lateinit var liststring:ArrayList<String>
    lateinit var adapter: ArrayAdapter<String>
```

MainActivity.kt

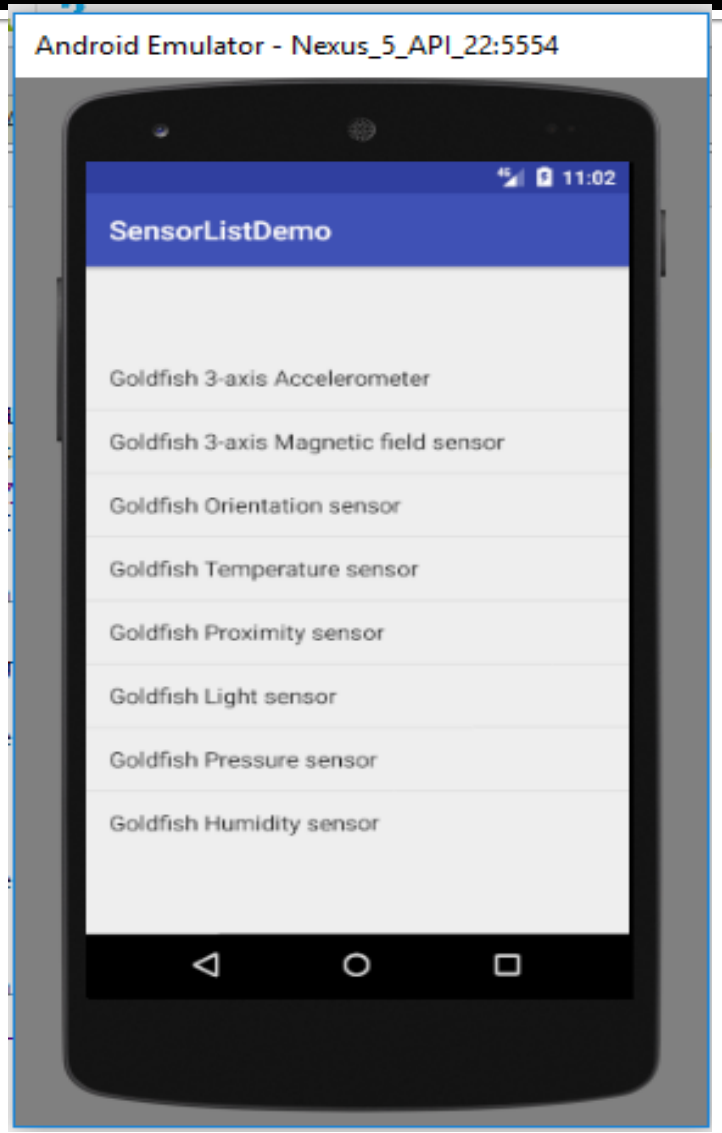
```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        listView = findViewById<ListView>(R.id.lv1)
        liststring = ArrayList()
        sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
        listsensor = sensorManager.getSensorList(Sensor.TYPE_ALL)
        for (i in listsensor.indices) {
            liststring.add(listsensor[i].name)
        }
        adapter = ArrayAdapter(this@MainActivity,
            android.R.layout.simple_list_item_1, liststring
        )
        listView.adapter = adapter
    }
}
```

Hands on Example – 1- Sensor List

Sample Output

This code is run through Emulator. ListView shows the available sensors from the Emulator. You can run this using your real device.

Refer : ShowSensorsList

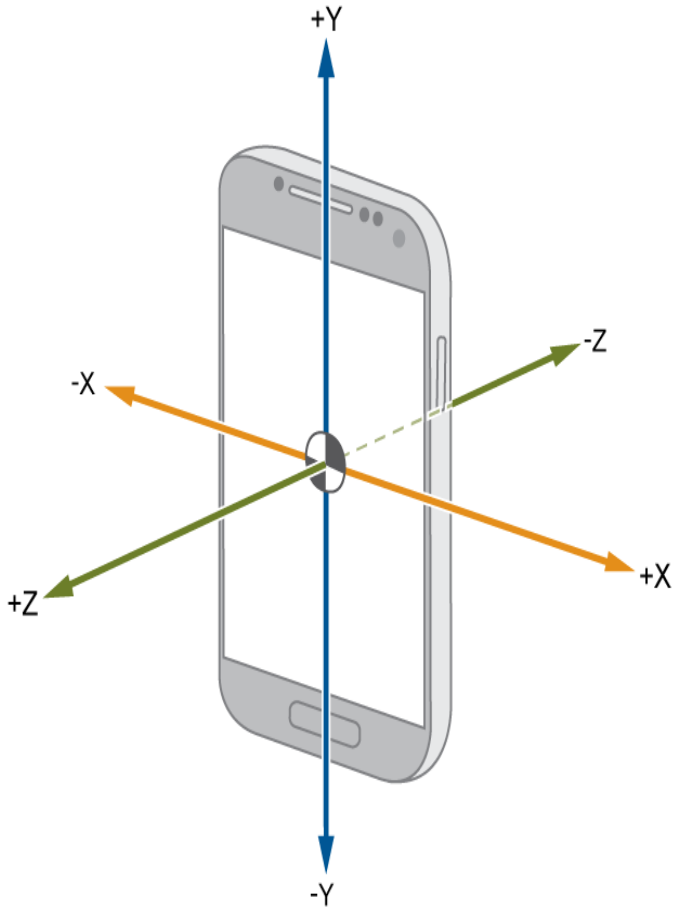


Main Point 1

Android provides SensorManager and Sensor classes to use the sensors in our application. In order to use sensors, first thing you need to do is to instantiate the object of SensorManager class. Then get the sensor event information while data changes through sensor event listener. ***Science of Consciousness:*** Transcendental Consciousness provides the most basic and useful foundation for developing any skill, because regular experience of this field stimulates the flow of thought and action from the total potential of our creativity and intelligence.

Hands on Example 2 - TYPE_ACCELEROMETER

- This example illustrates to know the x, y, z axis position on movements and if you shake your device fast, will play a sound.
- Create an xml file with one TextView component and configure id .
- Write your logic in MainActivity.java to play music and display the Coordinate position in the TextView.
- Refer : AccelerometerMovement



MainActivity.kt

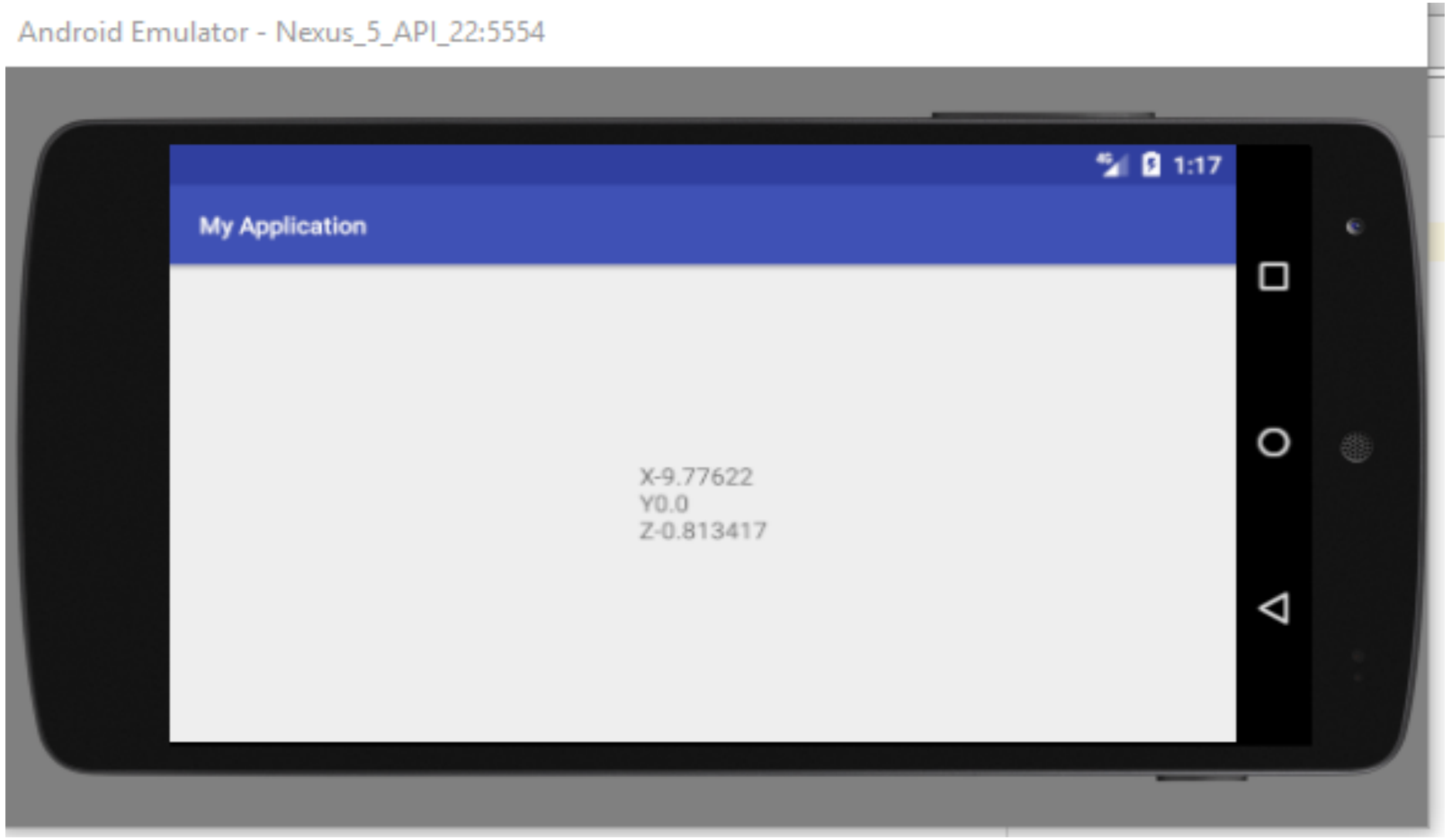
```
class MainActivity : AppCompatActivity(), SensorEventListener {
    lateinit  var sensor:Sensor
    //help us manage sensor components
    lateinit var sm:SensorManager
    lateinit var displayReading:TextView
    lateinit var mPlayer:MediaPlayer

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        sm = getSystemService(SENSOR_SERVICE) as SensorManager
        //select the sensor we wish to use
        sensor = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
        displayReading = findViewById(R.id.display_reading) as TextView
        mPlayer = MediaPlayer.create(this, R.raw.iphone)
    }
```

MainActivity.kt

```
// Register your Sensor Manager
    override fun onResume() {
        super.onResume()
        sm.registerListener(this, sensor, SensorManager.SENSOR_DELAY_NORMAL)
    }
    // Unregister your Sensor Manager
     override fun onPause() {
        super.onPause()
        sm.unregisterListener(this)    }
    override fun onSensorChanged(event:SensorEvent) {
        displayReading.setText("X" + event.values[0] + "\nY" + event.values[1] + "\nZ"
         + event.values[2])
        if (event.values[0] > 10){
            mPlayer.start()
        }    }
    // Called when the accuracy of a sensor has changed. We are not going to make use of this.
    override fun onAccuracyChanged(arg0:Sensor, arg1:Int) {
    }
}
```

Sample Output

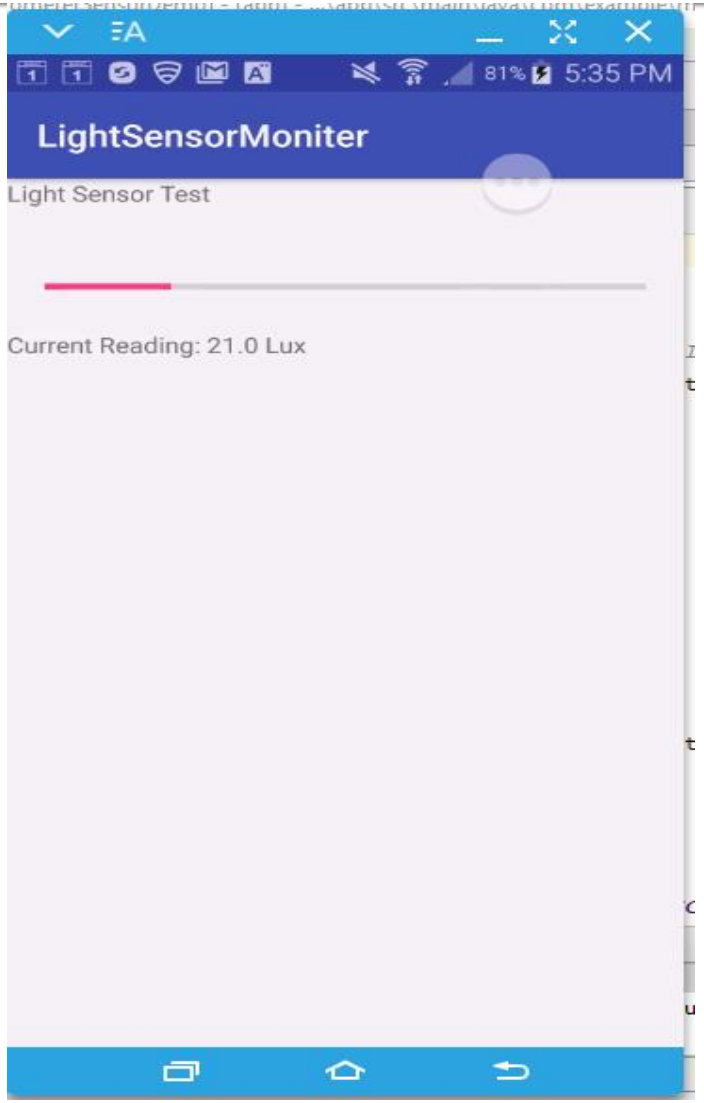


Hands on Example 3 - TYPE_LIGHT

- Measures the ambient light level (illumination) in lx.
- lx stands for lux.
- Useful for controlling screen brightness.
- Many mobile having Auto brightness mode function, this function work on light sensor that will adjust screen brightness as per light intensity.
- In this example we are reading light intensity value and display with progress bar.
- If you take your device from light to dark or dark to light, able to see the changes on Progress Bar.

Problem Requirement

- Design your layout with two TextView components and Progress Bar.
- Your MainActivity.kt needs to deal with TYPE_LIGHT sensor and show the updated lx value in the Progress bar and in TextView.
- Refer : LightLuxSensor



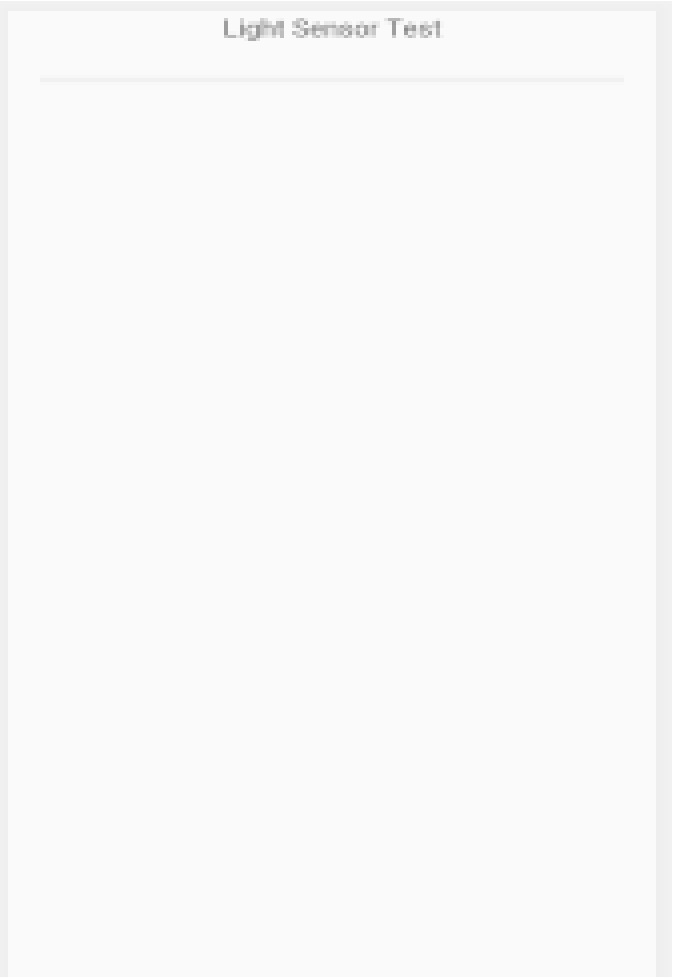
ProgressBar UI

ProgressBar UI

```
<ProgressBar
        android:id="@+id/lightmeter"
style="?android:attr/progressBarStyleHorizontal"
        android:layout_width=“match_parent"
        android:layout_height="wrap_content"
        android:max="100"
        android:padding="20dp"
        android:progress="0" />
```

android:progress : determine the amount of progress. This example set it as 0.

android:max : Progress bar get full once it reaches to 100.



MainActivity.kt

```
class MainActivity : AppCompatActivity(), SensorEventListener {
    private var sensorManager: SensorManager? = null
    private var lightSensor: Sensor?=null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        // implement sensor manager
        sensorManager = getSystemService(Context.SENSOR_SERVICE) as
                                SensorManager?
        lightSensor =         sensorManager!!.getDefaultSensor(Sensor.TYPE_LIGHT)
    }
    // Register your Sensor Manager
    override fun onResume() {
        super.onResume()
        sensorManager!!.registerListener(this, lightSensor, SensorManager.SENSOR_DELAY_NORMAL)
    }
```

MainActivity.kt

```
// Unregister your Sensor Manager
    override fun onPause() {
        super.onPause()
        sensorManager!!.unregisterListener(this)
    }
    // get sensor update and reading
    override fun onSensorChanged(event: SensorEvent) {
      if (event.sensor.type == Sensor.TYPE_LIGHT){
            val currentReading = event.values[0]
            lightmeter.progress = currentReading.toInt()
            reading.text = ("Current Reading: "
                    + (currentReading).toString() + " Lux")
        }  }
    override fun onAccuracyChanged(sensor:Sensor, i:Int) {     }}
```

Hands on Example 4 - TYPE_LIGHT

- Problem Requirement
  - If your device gets darkness, play music automatically.
  - Refer :LightOnOffMusic

MainActivity.kt

```
class MainActivity : AppCompatActivity(),SensorEventListener {
      private var sensor:Sensor? = null
       private var sm:SensorManager? = null
       lateinit  var mp:MediaPlayer
// Declare the boolean flag to know the player is running or not
    var flag = false
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        sm = getSystemService(Context.SENSOR_SERVICE) as SensorManager?
        sensor = sm!!.getDefaultSensor(Sensor.TYPE_LIGHT)
       // Check device has the requested Sensor or not
        if(sensor==null){
            Toast.makeText(this,"Your device has no Sensor.TYPE_LIGHT",Toast.LENGTH_LONG).show()    }}
```

MainActivity.kt

```
// Register your Sensor Manager
    override fun onResume() {
        super.onResume()
        sm!!.registerListener(this, sensor, SensorManager.SENSOR_DELAY_NORMAL)
    }
    // Unregister your Sensor Manager
    override fun onPause() {
        super.onPause()
        sm!!.unregisterListener(this)
    }
 override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {
// Nothing
}
```

MainActivity.kt

```
override fun onSensorChanged(event: SensorEvent?) {
        mp = MediaPlayer.create(applicationContext, R.raw.iphone)
        if (event!!.values!![0] < 20 )
        { // < 20 use for dark
            try   {
            mp.setOnCompletionListener(OnCompletionListener { mp.release() })
                mp.start()
            }
            catch (e: Exception) {
                e.printStackTrace()
            }
        }
    }
```

Main Point 2

TYPE_ACCELEROMETER is useful to know x, y, z axis position on movements and if you shake your device fast. TYPE_LIGHT measures the ambient light level (illumination) in lx. From these sensors, when shake the mobile or less light in the environment the device will play the music automatically to show its action. ***Science of Consciousness:*** Every action has a reaction. Established in Being, one performs right action like how sensors react depends on the motion and environment changes.

UNITY CHART

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

Support of nature

1. *Capturing and using motion, environmental, and position information and events provides a wealth of developmental possibilities in Android programming*
2. *Just as Creative Intelligence is Vigilant and Resourceful: Applying inner wealth through increasing alertness.*

---

3. **Transcendental Consciousness**: *TC is the home of all the impulses of natural law, of creative intelligence.*
4. ***Impulses within the Transcendental field:*** *These impulses are responsible for organizing the whole infinitely diverse universe.*
5. ***Wholeness moving within Itself****: In Unity Consciousness, the emergence the structure of pure knowledge as appreciated as a self-referral activity of consciousness interacting with itself.*