LESSON - 10

Sensors

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 - TYPE_LIGHT

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, relational vector sensors, and gyroscopes.
- **Environmental Sensors**: These sensors are used to measure environmental parameters such as temperature, pressure, humidity and so on.
- **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/s^2 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().
- **Sensor**
  - This class is used to retrieve a list of Sensors available in the devices.
- **SensorEvent**
  - This class stores information about the sensor type, sensor data, and so on.
- **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 you register your listener in the onResume() method and de-register it in the onPause() method.

Sensor Implementation Structure

```
public class SensorActivity extends Activity, implements SensorEventListener {
    private final SensorManager mSensorManager;
    private final Sensor mAccelerometer;

    public SensorActivity() {
        mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
        mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }

    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
    }

    protected void onPause() {
        super.onPause();
        mSensorManager.unregisterListener(this);
    }

    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    public void onSensorChanged(SensorEvent event) {
    }
}
```

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.java

MainActivity.java

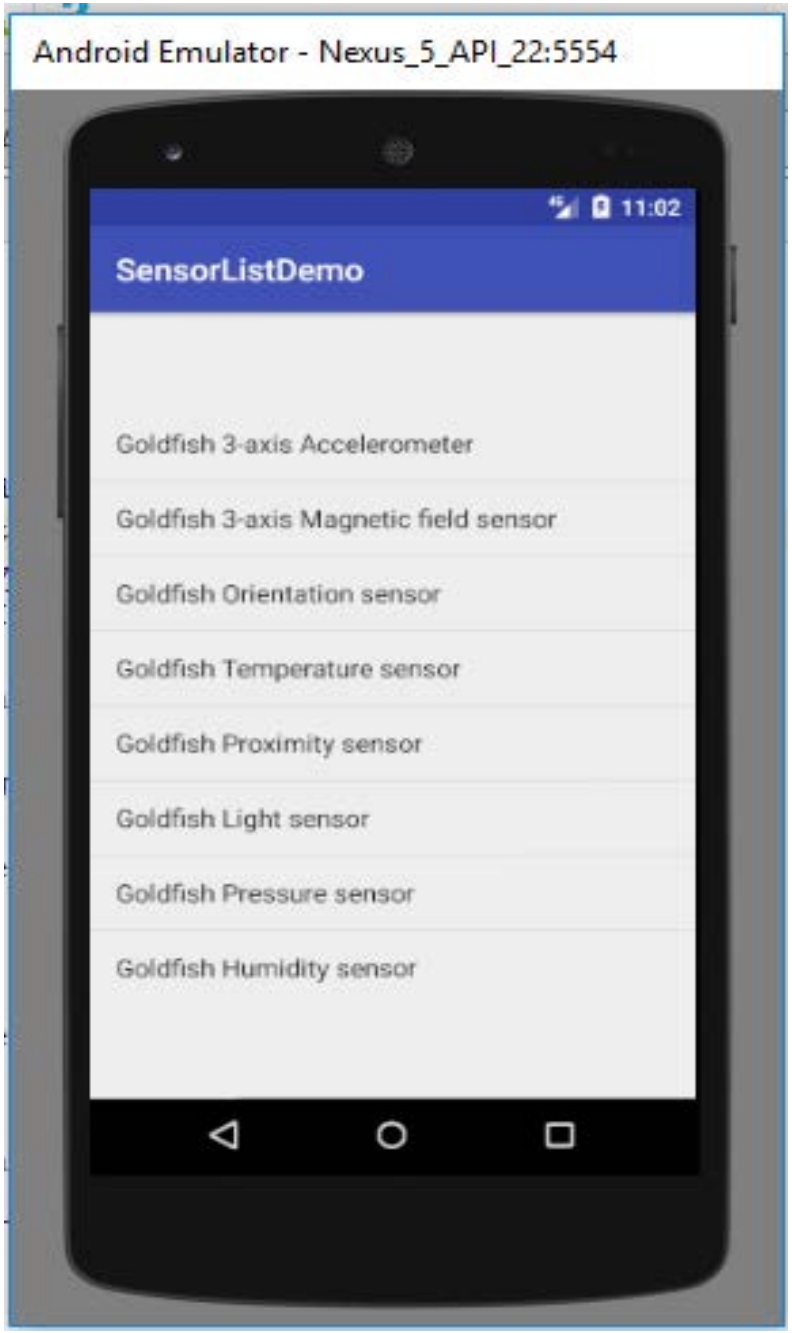
```
public class MainActivity extends AppCompatActivity {
    ListView listView ;
    SensorManager sensorManager ;
    List<Sensor> listsensor;
    List<String> liststring ;
    ArrayAdapter<String> adapter ;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        listView = (ListView)findViewById(R.id.lv1);
```

MainActivity.java

```
liststring = new ArrayList<String>();

        sensorManager =
(SensorManager)getSystemService(Context.SENSOR_SERVICE);

        listsensor = sensorManager.getSensorList(Sensor.TYPE_ALL);

        for(int i=0; i<listsensor.size(); i++){

            liststring.add(listsensor.get(i).getName());
        }

        adapter = new ArrayAdapter<String>(MainActivity.this,
                android.R.layout.simple_list_item_1,liststring
        );

        listView.setAdapter(adapter);

    }
}
```

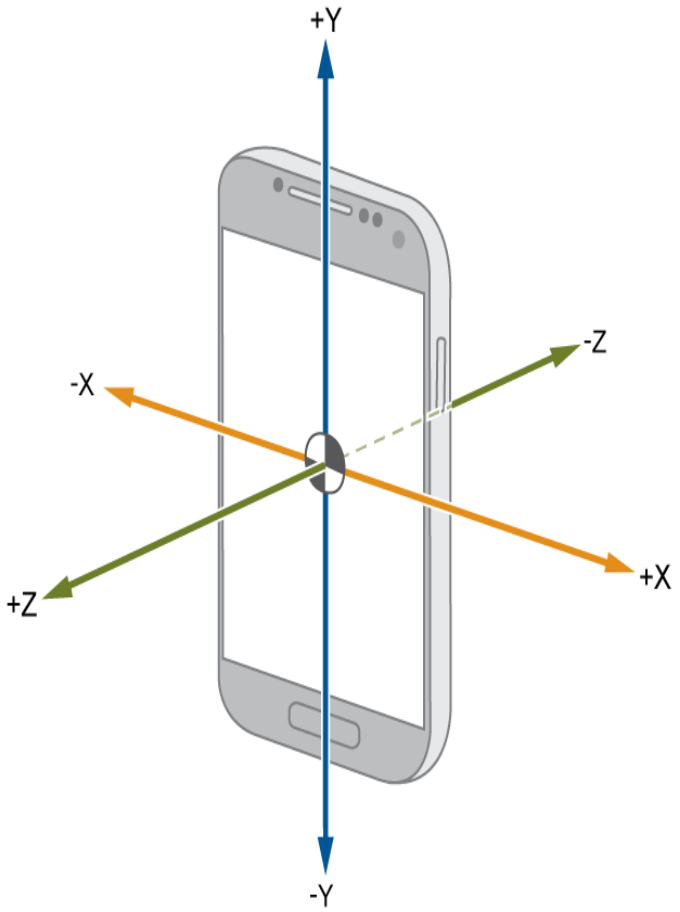
Sample Output

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



Hands on Example - **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.



MainActivity.java

```
public class MainActivity extends AppCompatActivity implements
SensorEventListener {
    //this class help select a particular sensor
    Sensor sensor;
    //help us manage sensor components
    SensorManager sm;

    TextView displayReading;
    MediaPlayer mPlayer;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //setup a sensor service
        sm = (SensorManager)getSystemService(SENSOR_SERVICE);

        //select the sensor we wish to use
        sensor = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        displayReading = (TextView)findViewById(R.id.display_reading);
        mPlayer = MediaPlayer.create(this, R.raw.iphone);
    }
```

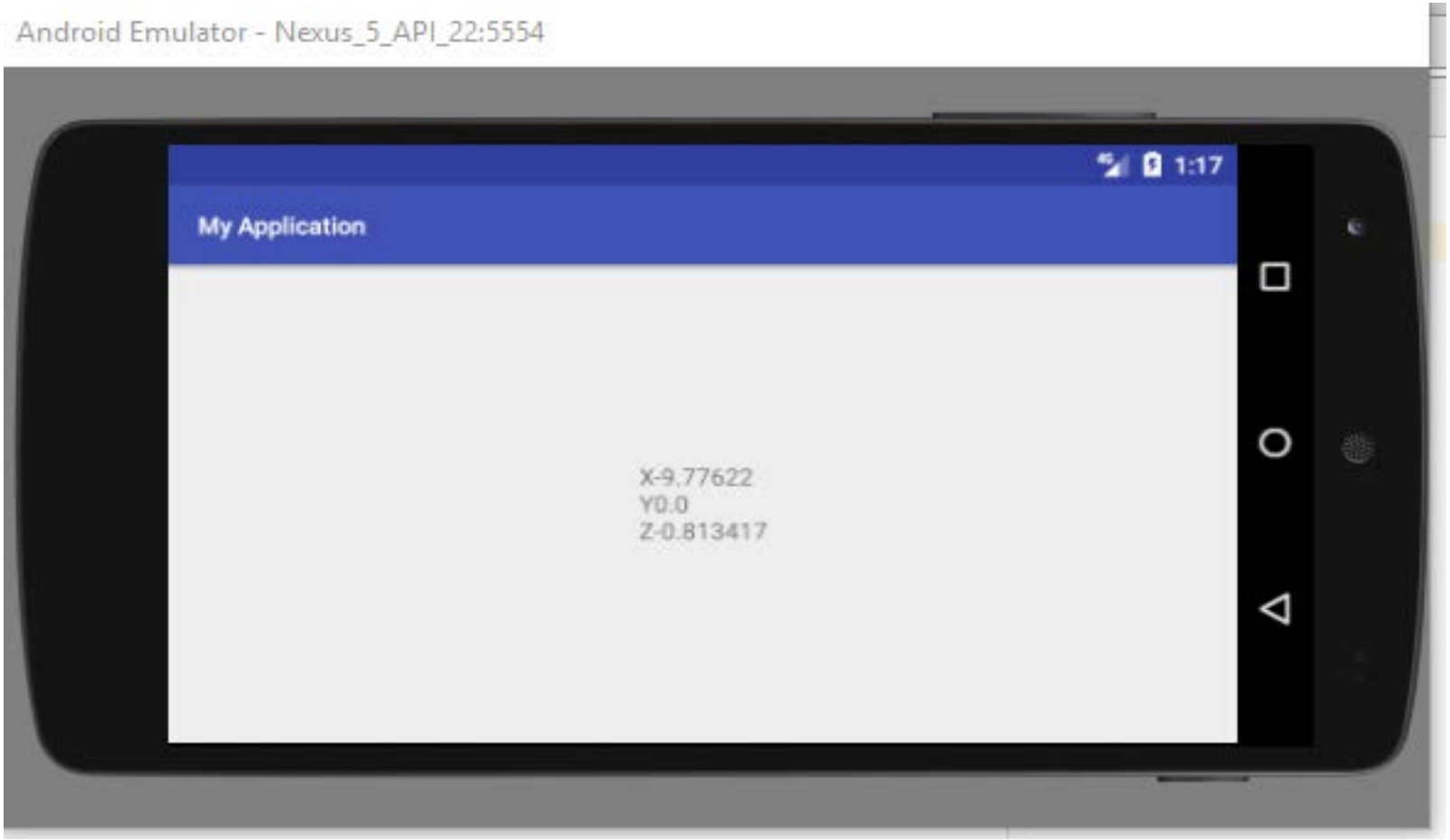
MainActivity.java

```
// Register your Sensor  Manager
@Override
protected void onResume() {
    super.onResume();
sm.registerListener(this,sensor,SensorManager.SENSOR_
DELAY_NORMAL);
}

// Unregister your Sensor Manager
@Override
protected void onPause() {
    super.onPause();
    sm.unregisterListener(this);
}
```

MainActivity.java

```
@Override
    public void onSensorChanged(SensorEvent event) {
       displayReading.setText("X"+event.values[0]+"\nY"+
            event.values[1]+"\nZ“+event.values[2]);

        if(event.values[0]>20)
        {
            mPlayer.start();
        }
    }
   /* Called when the accuracy of a sensor has changed. It will be called only one time. Every sensors is assigned with integer accuracy. */
    @Override
    public void onAccuracyChanged(Sensor arg0, int arg1) {

    }
}
```

Sample Output

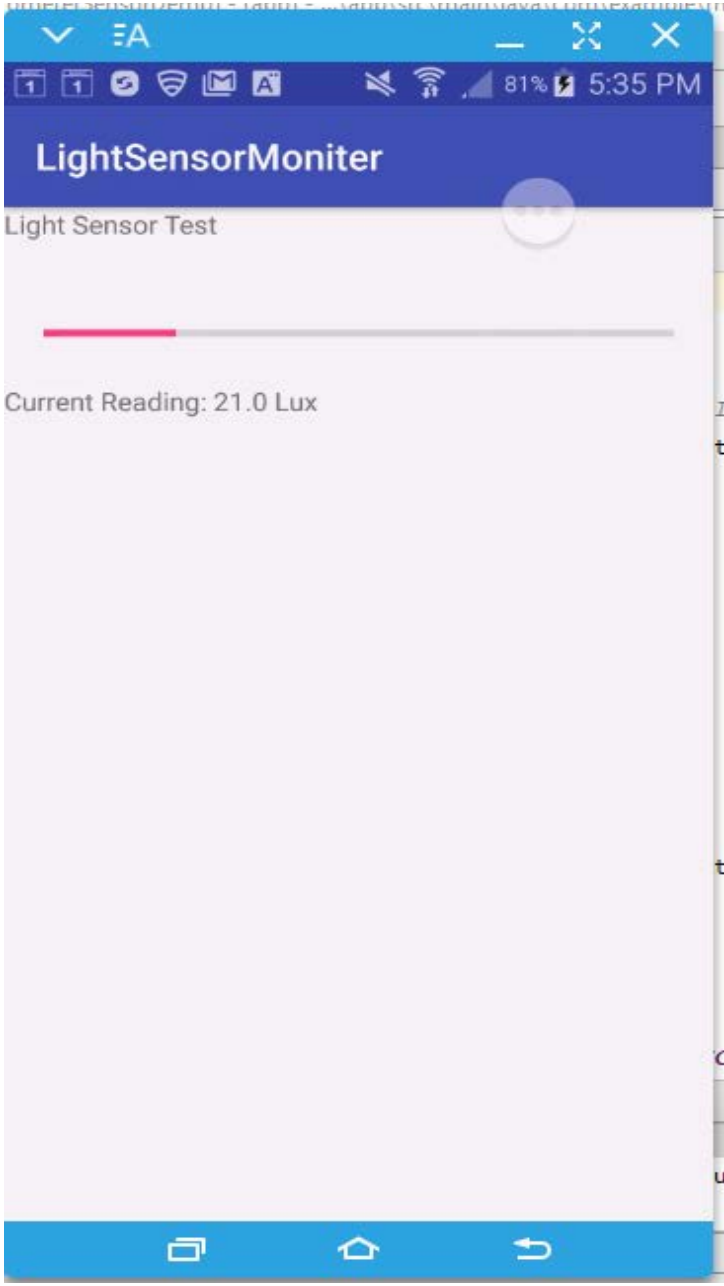


Hands on Example for TYPE_LIGHT

- 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.java needs to deal with TYPE_LIGHT sensor, and show the updated lx value in the Progress bar and in TextView.



MainActivity.java

```
public class MainActivity extends AppCompatActivity
implements SensorEventListener {
    ProgressBar lightMeter;
    TextView tvMaxValue, tvReader;
    SensorManager sensorManager;
    Sensor lightSensor;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // Load control
    lightMeter = (ProgressBar) findViewById(R.id.lightmeter);
    tvMaxValue = (TextView) findViewById(R.id.max);
    tvReader = (TextView) findViewById(R.id.reading);
```

```
// implement sensor manager
    sensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
    lightSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
    // check sensor available in devise. if available then get reading
    if (lightSensor == null) {
        Toast.makeText(getApplicationContext(), "No Sensor",
                Toast.LENGTH_SHORT).show();
    }
}
```

```
// Register your Sensor  Manager
@Override
protected void onResume() {
    super.onResume();
sensorManager.registerListener(this,lightSensor,SensorM
anager.SENSOR_DELAY_NORMAL);
}

// Unregister your Sensor Manager
@Override
protected void onPause() {
    super.onPause();
    sensorManager.unregisterListener(this);
}
```

```
@Override
    public void onSensorChanged(SensorEvent event) {
        if (event.sensor.getType() == Sensor.TYPE_LIGHT) {
            float currentReading = event.values[0];
            lightMeter.setProgress((int) currentReading);
            tvReader.setText("Current Reading: "
                    + String.valueOf(currentReading) + " Lux");
        }
    }

@Override
    public void onAccuracyChanged(Sensor sensor, int i) {
    }
}
```