Android Sensors

In this Lecture, you will learn:

- Sensor Manager
- Listing All Sensors
- Reading Sensor Data

Android devices have built-in sensors that measure motion, orientation, and other environmental condition. The android platform supports three broad categories of sensors:

Motion Sensors
Environmental sensors
Position sensors

Some of the sensors are hardware based and some are software based sensors. 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 components and derive their data from one or more of the hardware-based sensors. The linear acceleration sensor and the gravity sensor are examples of software-based sensors.

Sensor events

Monitoring sensor events is how you acquire raw sensor data. A sensor event occurs every time a sensor detects a change in the parameters it is measuring. A sensor event provides you with four pieces of information: the name of the sensor that triggered the event, the timestamp for the event, the accuracy of the event, and the raw sensor data that triggered the event.

SensorManager

Android allows to access the data from these sensors and use it in application. Some of the important classes to manage sensors are **SensorManager and Sensor.** To use sensors, instantiate the object of SensorManager class. It can be achieved as follows.

```
SensorManager sensorManager =
(SensorManager)this.getSystemService(SENSOR_SERVICE);
```

Now you can instantiate any sensor by calling getDefaultSensor() method of the SensorManager class. Before getting data from the sensor, it is better to check if the required sensor is available on the device.

```
Sensor sensorGyroscope =
sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
if (sensorGyroscope!=null){
    Toast.makeText(this, "sensor gyro init",
            Toast.LENGTH_LONG).show();
}else{
    Toast.makeText(this, "sensor gyro not available",
            Toast.LENGTH_LONG).show();
}
```

Monitoring Sensor Events

If the sensor is available and the object is initialized, register its listener and override two methods which are onAccuracyChanged and onSensorChanged.

```
sensorManager.registerListener(new SensorEventListener() {
                                @Override
                                public void
onSensorChanged(SensorEvent event) {
                                }
                                @Override
                                public void onAccuracyChanged(Sensor
sensor, int accuracy) {
                                }
                            },
        sensorLight,
        SensorManager.SENSOR_DELAY_NORMAL);
```

Getting list of supported sensors

You can get a list of sensors supported by your device by calling the getSensorList method, which will return a list of sensors containing their name and version number and other information.

```
List<Sensor> availSensor =
sensorManager.getSensorList(Sensor.TYPE_ALL);
String sensorType = "";
for(Sensor sensor: availSensor){
    sensorType+= sensor.getStringType();
}
Toast.makeText(this, sensorType,Toast.LENGTH_LONG).show();
```

Reading Sensor Data

Different sensors measure their specific kinds of measurements. An acceleration sensor measures the acceleration applied to the device, including the force of gravity.

```
public void initAccelerometer(){
    Sensor sensorAccelerometer =
mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    mSensorManager.registerListener(new SensorEventListener() {
        @Override
        public void onSensorChanged(SensorEvent event) {
            mTextViewSensorAccelerometer.setText("x:
"+event.values[0]+"\ny: "+event.values[1]+"\nz:
"+event.values[2]);
        }
        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy)
{
        }
    }
    ,sensorAccelerometer, SensorManager.SENSOR_DELAY_NORMAL
    );
}
```

Using Google Play filters to target specific sensor configurations

If you are publishing your application on Google Play you can use the <uses-feature> element in your manifest file to filter your application from devices that do not have the appropriate sensor configuration for your application. The following example filters devices that do not have an accelerometer:

```
<uses-feature android:name="android.hardware.sensor.accelerometer"
              android:required="true" />
```