Sensors Overview

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.

- 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. For example, you can use the sensor framework to do the following:

- Determine which sensors are available on a device.
- Determine an individual sensor's capabilities, such as its maximum range, manufacturer, power requirements, and resolution.
- Acquire raw sensor data and define the minimum rate at which you acquire sensor data.
- Register and unregister sensor event listeners that monitor sensor changes.

This topic provides an overview of the sensors that are available on the Android platform. It also provides an introduction to the sensor framework.

Introduction to Sensors

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. Table 1 summarizes the sensors that are supported by the Android platform.

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.

Table 1. Sensor types supported by the Android platform.

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` package and includes the following classes and interfaces:

`SensorManager`

> You can use this class to create an instance of the sensor service. This class 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.

`Sensor`

Sensor	Type	Description	Common Uses
TYPE_ACCELEROMETER	Hardware	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_AMBIENT_TEMPERATURE	Hardware	Measures the ambient room temperature in degrees Celsius (°C). See note below.	Monitoring air temperatures.
TYPE_GRAVITY	Software or Hardware	Measures the force of gravity in m/s² that is applied to a device on all three physical axes (x, y, z).	Motion detection (shake, tilt, etc.).
TYPE_GYROSCOPE	Hardware	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z).	Rotation detection (spin, turn, etc.).
TYPE_LIGHT	Hardware	Measures the ambient light level (illumination) in lx.	Controlling screen brightness.
TYPE_LINEAR_ACCELERATION	Software or Hardware	Measures the acceleration force in m/s² that is applied to a device on all three physical axes (x, y, and z), excluding the force of gravity.	Monitoring acceleration along a single axis.
TYPE_MAGNETIC_FIELD	Hardware	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in μT.	Creating a compass.

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.

TYPE_ORIENTATION	Software	Measures degrees of rotation that a device makes around all three physical axes (x, y, z). As of API level 3 you can obtain the inclination matrix and rotation matrix for a device by using the gravity sensor and the geomagnetic field sensor in conjunction with the getRotationMatrix() method.	Determining device position.
TYPE_PRESSURE	Hardware	Measures the ambient air pressure in hPa or mbar.	Monitoring air pressure changes.
TYPE_PROXIMITY	Hardware	Measures the proximity of an object in cm relative to the view screen of a device. This sensor is typically used to determine whether a handset is being held up to a person's ear.	Phone position during a call.
TYPE_RELATIVE_HUMIDITY	Hardware	Measures the relative ambient humidity in percent (%).	Monitoring dewpoint, absolute, and relative humidity.
TYPE_ROTATION_VECTOR	Software or Hardware	Measures the orientation of a device by providing the three elements of the device's rotation vector.	Motion detection and rotation detection.
TYPE_TEMPERATURE	Hardware	Measures the temperature of the device in degrees Celsius (°C). This sensor implementation varies across devices and this sensor was replaced with theTYPE_AMBIENT_TEMPERATURE sensor in API Level 14	Monitoring temperatures.

`SensorEvent`

The system uses this class to create a sensor event object, which provides information about a sensor event. A sensor event object includes the following information: the raw sensor data, the type of sensor that generated the event, the accuracy of the data, and the timestamp for the event.

`SensorEventListener`

You can use this interface to create two callback methods that receive notifications (sensor events) when sensor values change or when sensor accuracy changes.

In a typical application you use these sensor-related APIs to perform two basic tasks:

- **Identifying sensors and sensor capabilities**

  Identifying sensors and sensor capabilities at runtime is useful if your application has features that rely on specific sensor types or capabilities. For example, you may want to identify all of the sensors that are present on a device and disable any application features that rely on sensors that are not present. Likewise, you may want to identify all of the sensors of a given type so you can choose the sensor implementation that has the optimum performance for your application.

- **Monitor 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.

Monitoring Sensor Events

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

- **A sensor's accuracy changes.**

  In this case the system invokes the `onAccuracyChanged()` method, providing you with a reference to the `Sensor` object that changed and the new accuracy of the sensor. Accuracy is represented by one of four status
  constants: `SENSOR_STATUS_ACCURACY_LOW`, `SENSOR_STATUS_ACCURACY_MEDIUM`,`SENSOR_STATUS_ACCURACY_HIGH`, or `SENSOR_STATUS_UNRELIABLE`.

- **A sensor reports a new value.**

In this case the system invokes the `onSensorChanged()` method, providing you with a `SensorEvent` object. A `SensorEvent` object contains information about the new sensor data, including: the accuracy of the data, the sensor that generated the data, the timestamp at which the data was generated, and the new data that the sensor recorded.

The following code shows how to use the `onSensorChanged()` method to monitor data from the light sensor. This example displays the raw sensor data in a `TextView` that is defined in the main.xml file as `sensor_data`.

```
public class SensorActivity extends Activity implements SensorEventListener {
  private SensorManager mSensorManager;
  private Sensor mLight;

  @Override
  public final void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    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) {
    // The light sensor returns a single value.
    // Many sensors return 3 values, one for each axis.
    float lux = event.values[0];
    // Do something with this sensor value.
  }

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

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

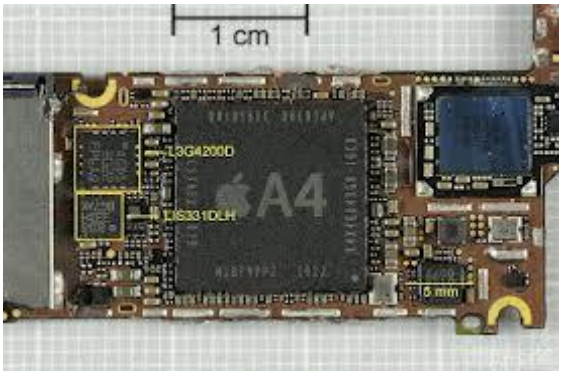
In this example, the default data delay (`SENSOR_DELAY_NORMAL`) is specified when the `registerListener()` method is invoked. The data delay (or sampling rate) controls the interval at which sensor events are sent to your application via the `onSensorChanged()` callback method. The default data delay is suitable for monitoring typical screen orientation changes and uses a delay of 200,000 microseconds. You can specify other data delays, such as `SENSOR_DELAY_GAME` (20,000 microsecond delay), `SENSOR_DELAY_UI` (60,000 microsecond delay), or `SENSOR_DELAY_FASTEST` (0 microsecond delay). As of Android 3.0 (API Level 11) you can also specify the delay as an absolute value (in microseconds).

The delay that you specify is only a suggested delay. The Android system and other applications can alter this delay. As a best practice, you should specify the largest delay that you can because the system typically uses a smaller delay than the one you specify (that is, you should choose the slowest sampling rate that still meets the needs of your application). Using a larger delay imposes a lower load on the processor and therefore uses less power.

There is no public method for determining the rate at which the sensor framework is sending sensor events to your application; however, you can use the timestamps that are associated with each sensor event to calculate the sampling rate over several events. You should not have to change the sampling rate (delay) once you set it. If for some reason you do need to change the delay, you will have to unregister and reregister the sensor listener.

It's also important to note that this example uses the `onResume()` and `onPause()` callback methods to register and unregister the sensor event listener. As a best practice you should always disable sensors you don't need, especially when your activity is paused. Failing to do so can drain the battery in just a few hours because some sensors have substantial power requirements and can use up battery power quickly. The system will not disable sensors automatically when the screen turns off

ACCELEROMETER AND GYROSCOPE :



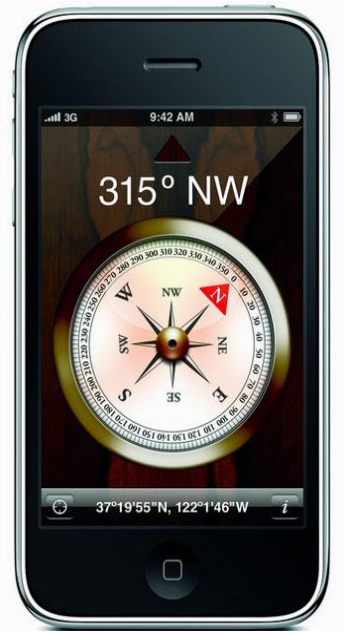
The purpose of gyroscope in a mobile is to detect the orientation of the mobile. On the other hand gyroscope is used to provide the information regarding the tracking rotation or the twist. In physics terms, almost every smartphone in the market has got some types of sensors in it. Especially the touch an accelerometer is used for measuring the linear acceleration and a gyroscope is used for measuring the angular rotational velocity. In short, these two sensors are used for measuring the rate of change of different things.

The accelerometer can only provide you information regarding the directional movement of the mobile but cannot tell you anything about the lateral orientation or tilt during. For this purpose you will need the gyroscope.

Both these sensors with assistance of each other. By using an accelerometer only, you will be able to produce a clean output but it will be very sluggish or it will be very noisy but will be very responsive. You will need to combine 3-axis accelerometer and the 3-axis gyro for producing an output that is both, clean and responsive.

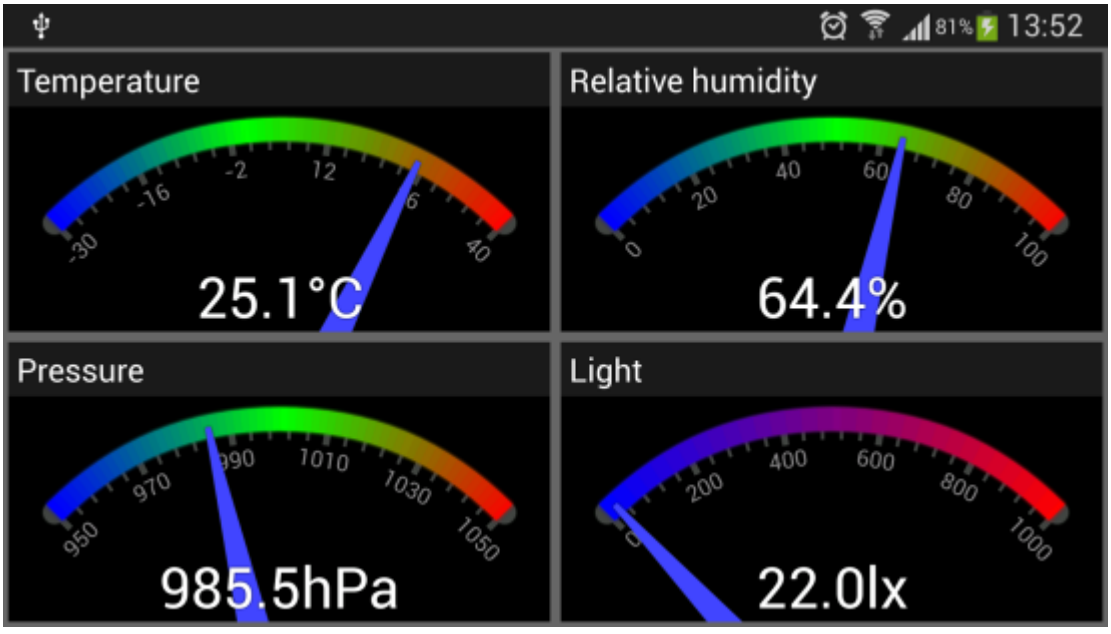
COMPASS OR MAGNETOMETER:

As the name of the sensor suggests, it is a simple digital compass which is based on a sensor called magnetometer and makes the mobile to work as a simple traditional compass It provides simple orientation in relation to the magnetic field of our Earth. This sensor is mostly used in the digital maps for letting you know that which way is North.



BAROMETER:

Higher-end phones have a built-in ***barometer*** – a sensor that can measure atmospheric pressure. Contrary to what you may suggest, it has nothing to do with weather. Instead, the data measured by it is used to determine how high the device is above sea level, to help the ***GPS*** chip inside the device get a faster lock by instantly delivering altitude data, which in turn results in improved ***GPS*** accuracy. On a related note, the Motorola XOOM and the Samsung Galaxy Nexus were among the first Android devices to feature this sensor. This sensor is found in only some selective latest mobiles.



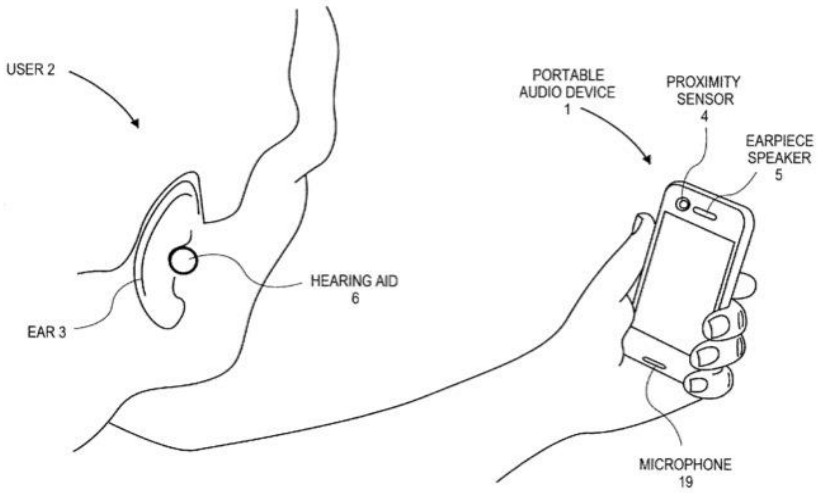
THE THERMOMETER

Some folks might remember that the *Samsung Galaxy S4* bragged with a ***thermometer*** for measuring ambient temperature. However, there's a ***thermometer*** in pretty much any smartphone, and some handsets might have more than one of them. The difference is that they're used to monitor the *temperature* inside the device and its battery. If a component is detected to be overheating, the system shuts itself down to prevent damage. And speaking of the Galaxy S4, it pioneered the use of an *air humidity sensor* in a smartphone.

BACK-ILLUMINATED SENSOR

Back-illuminated ***sensor*** is one of the new feature that every camera contains. It is a type of *digital image sensor* which changes or increase the light captured while capturing a photograph. Earlier it was designed for security cameras and astronomical purposes. *Sony* is the first company to implement this technology in 2009.

PROXIMITY:



The proximity sensor is used to find out that how much your phone close to your body. When you bring the mobile closed to your ear, the proximity sensor detects it and automatically closes the display for saving the battery. This thing also helps in avoiding the accidental touch cause by your ear while answering a phone call.

THE HEART RATE SENSOR

For the first time in the history of the smartphone, Galaxy S5 features the heart rate sensor, which enables users to monitor their physical information. This opens up a lot of opportunities in terms of the benefit that can be produced from the utilization of such information.

Users can check their heart rate through the S Health app. By following the instructions that the S Health provides, users simply need to gently press their finger against the heart rate sensor, which is located on the bottom of the camera on the back panel.

The Heart Rate Sensor consists of the Red LED and Pulse Sensor. First, the Red LED shoots the light to the user's skin. Then the pulse sensor measures the movement of the red blood cells of the capillaries, which is underneath the finger, according to the pulsation, to measure the heart rate by calculating the frequency of the wave per minute.

The measured heart rate sensor is displayed on smartphone, and heart rate measurements can be recorded and saved into the smartphone. Users can measure their heart rate before and after a workout to check out their health and workout status.

THE FINGER SCANNER

One other 'most mentioned' sensor of the Galaxy S5 is the Finger Scanner, which has improved the usability and the security of the smartphone.

The Finger Scanner of the Galaxy S5 supports not only a biometric screen locking feature, but also supports individual file locking features with 'Private Mode' and secure mobile payments.

For example, People tend to use relatively simple passwords, such as their birthday, which is relatively unsecure. Utilizing biometric information, users improve the security of their mobile purchasing. Maximizing opportunity and technology, Samsung has forged a partnership with Paypal, an internet mobile payment system, to designate passwords with fingerprints to make a mobile purchasing easier and more secure in 26 countries.

One of the main concerns of the finger scanner has been the personal data breach of fingerprints, because fingerprints tend to be saved as an image.

In the Galaxy S5, the distinctive characteristics of fingerprints are extracted and encrypted rather than saved as a full image. Then it is processed as a template through the secured zone of the device's chip. Moreover, every devices are encrypted uniquely, so no one can decode the information regarding the fingerprint, even Samsung itself.

Therefore, users' fingerprints are not saved in the device, so there is no danger of information disclosure.
Users can register 3 fingerprints, and these fingerprints can be recognized in either a perpendicular or parallel direction. To maintain the unique design of the Galaxy S series while mounting the Finger Scanner feature, it has applied the feature called Swype. Using the Swype method, users fingerprints can be recognized by swiping his/her finger from the bottom of the screen to the home button.

THE PEDOMETER

Used for counting the number of steps that the user has taken. Such data is usually obtained by the device's accelerometer, but a dedicated ***pedometer*** is a lot more accurate and power efficient. The *Google Nexus 5* is among the few phones that have a true pedometer built into them.

Detect Harmful Radiation

A ***sensor*** that you wouldn't expect to find on a smartphone is one capable of detecting harmful radiation. Yet there's a phone that sports one – the *Sharp Pantone 5*. Released only in Japan, it features a dedicated button which launches an app used to measure the current radiation level in the area.

RGB Light Sensor: Measure the red, green, blue and white intensity of the light source, used mainly to make adjustments in the Cameras.

Hall Sensor: Recognizes whether the cover is open or closed.

REFERENCES

1. https://developer.android.com/guide/topics/sensors/sensors_overview.html
2. https://testingmobileapps.wordpress.com/2016/02/17/smartphones-sensors-list/