**Descripción general de sensores**

La mayoría de los dispositivos con Android tienen sensores integrados que miden el movimiento, la orientación y diversas condiciones ambientales. Estos sensores son capaces de proporcionar datos sin procesar con alta precisión y exactitud, y son útiles para supervisar el movimiento o posicionamiento tridimensional del dispositivo, o si quieres supervisar los cambios en el entorno ambiental cerca de un dispositivo. Por ejemplo, un juego puede hacer un seguimiento de las lecturas del sensor de gravedad de un dispositivo para inferir gestos y movimientos complejos del usuario, como inclinación, agitación, rotación o balanceo. Del mismo modo, una aplicación meteorológica podría usar el sensor de temperatura y el sensor de humedad de un dispositivo para calcular e informar el punto de condensación, o una aplicación de viaje podría usar el sensor de campo geomagnético y el acelerómetro para informar la dirección de una brújula.

Consulta los siguientes recursos relacionados:

* [Sensores](https://developer.android.com/guide/topics/sensors?hl=es)
* [Sensores de movimiento](https://developer.android.com/guide/topics/sensors/sensors_motion?hl=es)
* [Sensores de posición](https://developer.android.com/guide/topics/sensors/sensors_position?hl=es)
* [Sensores ambientales](https://developer.android.com/guide/topics/sensors/sensors_environment?hl=es)
* [Reproducción de acelerómetro](https://github.com/googlesamples/android-AccelerometerPlay)
* [Demostraciones de API (SO: RotationVectorDemo)](https://android.googlesource.com/platform/development/+/master/samples/ApiDemos/src/com/example/android/apis/os/RotationVectorDemo.java)

La plataforma de Android admite tres amplias categorías de sensores:

* Sensores de movimiento  
  Estos sensores miden las fuerzas de aceleración y las fuerzas de rotación en tres ejes. Esta categoría incluye acelerómetros, sensores de gravedad, giroscopios y sensores del vector de rotación.
* Sensores ambientales  
  Estos sensores miden varios parámetros ambientales, como la temperatura y la presión del aire ambiental, la iluminación y la humedad. Esta categoría incluye barómetros, fotómetros y termómetros.
* Sensores de posición  
  Estos sensores miden la posición física de un dispositivo. Esta categoría incluye sensores de orientación y magnetómetros.

Puedes acceder a los sensores disponibles en el dispositivo y adquirir datos sin procesar del sensor mediante el marco de trabajo del sensor de Android. El marco de trabajo del sensor proporciona varias clases e interfaces que ayudan a realizar una amplia variedad de tareas relacionadas con el sensor. Por ejemplo, puedes usar el marco de trabajo del sensor para las siguientes tareas:

* Determina qué sensores están disponibles en un dispositivo.
* Determina las capacidades de un sensor individual, como su rango máximo, el fabricante, los requisitos de energía y la resolución.
* Adquiere datos sin procesar del sensor y define la velocidad mínima con la que adquirirás los datos del sensor.
* Registra y cancela el registro de oyentes de objetos de escucha de eventos que supervisan los cambios del sensor.

En este tema, se proporciona una descripción general de los sensores que están disponibles en la plataforma de Android. También se proporciona una introducción al marco de trabajo del sensor.

## **Introducción a los sensores**

El marco de trabajo del sensor de Android permite acceder a muchos tipos de sensores. Algunos de estos sensores se basan en hardware y otros en software. Los sensores basados en hardware son componentes físicos integrados en un dispositivo de mano o tablet. Para obtener los datos, miden directamente propiedades ambientales específicas, como la aceleración, la intensidad del campo geomagnético o el cambio angular. Los sensores basados en software no son dispositivos físicos, aunque imitan los sensores basados en hardware. Los sensores basados en software derivan sus datos de uno o más de los sensores basados en hardware y, a veces, se denominan sensores virtuales o sensores sintéticos. El sensor de aceleración lineal y el sensor de gravedad son ejemplos de sensores basados en software. En la tabla 1, se resumen los sensores que son compatibles con la plataforma de Android.

Solo unos pocos dispositivos con Android tienen todos los tipos de sensores. Por ejemplo, la mayoría de los dispositivos manuales y tablets tienen un acelerómetro y un magnetómetro, pero solo unos pocos dispositivos tienen barómetros o termómetros. Además, un dispositivo puede tener más de un sensor de un tipo determinado. Por ejemplo, un dispositivo puede tener dos sensores de gravedad, cada uno con un rango diferente.

**Tabla 1:** Tipos de sensores compatibles con la plataforma de Android

| **Sensor** | **Tipo** | **Descripción** | **Usos habituales** |
| --- | --- | --- | --- |
| [TYPE\_ACCELEROMETER](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_ACCELEROMETER) | Hardware | Mide en m/s 2 la fuerza de aceleración que se aplica a un dispositivo en los tres ejes físicos (x, y, z), incluida la fuerza de gravedad. | Detección de movimiento (agitación, inclinación, etc.). |
| [TYPE\_AMBIENT\_TEMPERATURE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_AMBIENT_TEMPERATURE) | Hardware | Mide la temperatura ambiente de la habitación en grados Celsius (°C). Consulta la siguiente nota. | Supervisión de la temperatura del aire. |
| [TYPE\_GRAVITY](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_GRAVITY) | Software o hardware | Mide en m/s2 la fuerza de gravedad que se aplica a un dispositivo en los tres ejes físicos (x, y, z). | Detección de movimiento (agitación, inclinación, etc.). |
| [TYPE\_GYROSCOPE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_GYROSCOPE) | Hardware | Mide en rad/s la velocidad de rotación de un dispositivo alrededor de cada uno de los tres ejes físicos (x, y, z). | Detección de rotación (agitación, giro, etc.). |
| [TYPE\_LIGHT](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_LIGHT) | Hardware | Mide el nivel de luz ambiental (iluminación) en lx. | Control del brillo de la pantalla. |
| [TYPE\_LINEAR\_ACCELERATION](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_LINEAR_ACCELERATION) | Software o hardware | Mide en m/s2 la fuerza de aceleración que se aplica a un dispositivo en los tres ejes físicos (x, y, z), excluyendo la fuerza de gravedad. | Supervisión de la aceleración a lo largo de un solo eje. |
| [TYPE\_MAGNETIC\_FIELD](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_MAGNETIC_FIELD) | Hardware | Mide el campo geomagnético ambiental de los tres ejes físicos (x, y, z) en μT. | Creación de una brújula. |
| [TYPE\_ORIENTATION](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_ORIENTATION) | Software | Mide los grados de rotación de un dispositivo alrededor de los tres ejes físicos (x, y, z). A partir de la API nivel 3, puedes obtener la matriz de inclinación y la matriz de rotación de un dispositivo mediante el uso del sensor de gravedad y el sensor del campo geomagnético de un dispositivo usando el sensor de gravedad junto con el método [getRotationMatrix()](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#getRotationMatrix(float%5B%5D,%20float%5B%5D,%20float%5B%5D,%20float%5B%5D)). | Determinación de la posición del dispositivo. |
| [TYPE\_PRESSURE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_PRESSURE) | Hardware | Mide la presión del aire del ambiente en hPa o mbar. | Supervisa los cambios de la presión del aire. |
| [TYPE\_PROXIMITY](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_PROXIMITY) | Hardware | Mide en cm la proximidad de un objeto con respecto a la pantalla de visualización de un dispositivo. Este sensor en general se usa para determinar si un dispositivo manual se está sosteniendo cerca del oído de una persona. | Posición del teléfono durante una llamada. |
| [TYPE\_RELATIVE\_HUMIDITY](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_RELATIVE_HUMIDITY) | Hardware | Mide en valor de porcentaje (%) la humedad relativa del ambiente. | Supervisa el punto de condensación, la humedad absoluta y la humedad relativa. |
| [TYPE\_ROTATION\_VECTOR](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_ROTATION_VECTOR) | Software o hardware | Mide la orientación de un dispositivo mediante los tres elementos del vector de rotación del dispositivo. | Detección de movimiento y detección de rotación. |
| [TYPE\_TEMPERATURE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_TEMPERATURE) | Hardware | Mide la temperatura del dispositivo en grados Celsius (°C). La implementación de este sensor varía según el dispositivo; en la API nivel 14 se reemplazó por el sensor [TYPE\_AMBIENT\_TEMPERATURE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_AMBIENT_TEMPERATURE). | Supervisión de temperaturas. |

### **Marco de trabajo del sensor**

Puedes acceder a estos sensores y adquirir datos del sensor sin procesar con el marco de trabajo del sensor de Android. El marco de trabajo del sensor es parte del paquete [android.hardware](https://developer.android.com/reference/android/hardware/package-summary?hl=es) e incluye las siguientes clases e interfaces:

[SensorManager](https://developer.android.com/reference/android/hardware/SensorManager?hl=es)

Puedes usar esta clase para crear una instancia del servicio del sensor. Esta clase proporciona varios métodos para acceder a sensores y escucharlos, registrar y cancelar el registro de objetos de escucha de sensores de eventos y adquirir información de orientación. También proporciona varias constantes del sensor que se usan para informar la exactitud del sensor, definir las velocidades de adquisición de datos y calibrar sensores.

[Sensor](https://developer.android.com/reference/android/hardware/Sensor?hl=es)

Puedes usar esta clase para crear una instancia de un sensor específico. Esta clase proporciona varios métodos que permiten determinar las capacidades de un sensor.

[SensorEvent](https://developer.android.com/reference/android/hardware/SensorEvent?hl=es)

El sistema usa esta clase para crear un objeto de evento de sensor, que proporciona información sobre un evento del sensor. Un objeto de evento de sensor incluye la siguiente información: los datos sin procesar del sensor, el tipo de sensor que generó el evento, la exactitud de los datos y la marca de tiempo del evento.

[SensorEventListener](https://developer.android.com/reference/android/hardware/SensorEventListener?hl=es)

Puedes usar esta interfaz para crear dos métodos de devolución de llamada que reciben notificaciones (eventos del sensor) cuando cambian los valores del sensor o cuando cambia la exactitud del sensor.

En una aplicación típica, estas API relacionadas con sensores se usan para realizar dos tareas básicas:

* **Identificación de sensores y capacidades de los sensores**La identificación de sensores y capacidades de sensores en el tiempo de ejecución es útil si la aplicación tiene características que dependen de tipos o capacidades de sensores específicos. Por ejemplo, es posible que quieras identificar todos los sensores que están presentes en un dispositivo e inhabilitar las funciones de la aplicación que dependen de sensores que no están presentes. Del mismo modo, es posible que quieras identificar todos los sensores de un tipo determinado para que puedas elegir la implementación del sensor que tenga el rendimiento óptimo para tu aplicación.
* **Supervisión de los eventos del sensor**La supervisión de los eventos del sensor es la manera en la que adquieres los datos sin procesar del sensor. Un evento de sensor ocurre cada vez que un sensor detecta un cambio en los parámetros que está midiendo. Un evento de sensor proporciona cuatro tipos de datos: el nombre del sensor que activó el evento, la marca de tiempo del evento, la exactitud del evento y los datos sin procesar del sensor que activaron el evento.

### **Disponibilidad del sensor**

Si bien la disponibilidad del sensor varía de un dispositivo a otro, también puede variar de una versión de Android a otra. Esto se debe a que los sensores de Android se introdujeron en el transcurso de varias versiones de la plataforma. Por ejemplo, muchos sensores se introdujeron en Android 1.5 (API nivel 3), pero algunos no se implementaron ni estuvieron disponibles hasta Android 2.3 (API nivel 9). Asimismo, se introdujeron varios sensores en Android 2.3 (API nivel 9) y Android 4.0 (API nivel 14). Dos sensores dejaron de estar disponibles y se reemplazaron por sensores más recientes y mejorados.

En la tabla 2, se resume la disponibilidad de los sensores en cada una de las plataformas. Solo se enumeran cuatro plataformas porque son las plataformas que involucraron cambios en los sensores. Los sensores enumerados como obsoletos seguirán estando disponibles en plataformas posteriores (siempre que el sensor esté presente en un dispositivo), en concordancia con la política de compatibilidad de versiones anteriores de Android.

**Tabla 2.** Disponibilidad de sensores por plataforma

| **Sensor** | **Android 4.0**  **(API nivel 14)** | **Android 2.3**  **(API nivel 9)** | **Android 2.2**  **(API nivel 8)** | **Android 1.5**  **(API nivel 3)** |
| --- | --- | --- | --- | --- |
| [TYPE\_ACCELEROMETER](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_ACCELEROMETER) | Sí | Sí | Sí | Sí |
| [TYPE\_AMBIENT\_TEMPERATURE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_AMBIENT_TEMPERATURE) | Sí | n/a | n/a | n/a |
| [TYPE\_GRAVITY](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_GRAVITY) | Sí | Sí | n/a | n/a |
| [TYPE\_GYROSCOPE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_GYROSCOPE) | Sí | Sí | n/a1 | n/a1 |
| [TYPE\_LIGHT](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_LIGHT) | Sí | Sí | Sí | Sí |
| [TYPE\_LINEAR\_ACCELERATION](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_LINEAR_ACCELERATION) | Sí | Sí | n/a | n/a |
| [TYPE\_MAGNETIC\_FIELD](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_MAGNETIC_FIELD) | Sí | Sí | Sí | Sí |
| [TYPE\_ORIENTATION](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_ORIENTATION) | Sí2 | Sí2 | Sí2 | Sí |
| [TYPE\_PRESSURE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_PRESSURE) | Sí | Sí | n/a1 | n/a1 |
| [TYPE\_PROXIMITY](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_PROXIMITY) | Sí | Sí | Sí | Sí |
| [TYPE\_RELATIVE\_HUMIDITY](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_RELATIVE_HUMIDITY) | Sí | n/a | n/a | n/a |
| [TYPE\_ROTATION\_VECTOR](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_ROTATION_VECTOR) | Sí | Sí | n/a | n/a |
| [TYPE\_TEMPERATURE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_TEMPERATURE) | Sí2 | Sí | Sí | Sí |

**1** Este tipo de sensor se agregó en Android 1.5 (API nivel 3), pero no estuvo disponible para Android 2.3 (API nivel 9).

**2** Este sensor está disponible, pero es obsoleto.

## **Identificación de sensores y capacidades de sensores**

El marco de trabajo del sensor de Android proporciona varios métodos que te permiten determinar con facilidad en el tiempo de ejecución qué sensores están en un dispositivo. La API también proporciona métodos que permiten determinar las capacidades de cada sensor, como el rango máximo, la resolución y los requisitos de potencia.

Para identificar los sensores que están en un dispositivo, primero debes obtener una referencia al servicio del sensor. Para ello, crea una instancia de la clase [SensorManager](https://developer.android.com/reference/android/hardware/SensorManager?hl=es) llamando al método [getSystemService()](https://developer.android.com/reference/android/content/Context?hl=es#getSystemService(java.lang.Class%3CT%3E)) y pasando el argumento [SENSOR\_SERVICE](https://developer.android.com/reference/android/content/Context?hl=es#SENSOR_SERVICE). Por ejemplo:

[Kotlin](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#kotlin)

[Java](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#java)

private SensorManager sensorManager;

...

sensorManager = (SensorManager) getSystemService(Context.SENSOR\_SERVICE);

Luego, puedes obtener una lista de cada sensor en un dispositivo llamando al método [getSensorList()](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#getSensorList(int)) y utilizando la constante [TYPE\_ALL](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_ALL). Por ejemplo:

[Kotlin](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#kotlin)

[Java](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#java)

List<Sensor> deviceSensors = sensorManager.getSensorList(Sensor.TYPE\_ALL);

Si quieres enumerar todos los sensores de un tipo determinado, puedes usar otra constante en lugar de [TYPE\_ALL](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_ALL), como [TYPE\_GYROSCOPE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_GYROSCOPE), [TYPE\_LINEAR\_ACCELERATION](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_LINEAR_ACCELERATION) o [TYPE\_GRAVITY](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_GRAVITY).

También puedes determinar si un tipo específico de sensor existe en un dispositivo mediante el método [getDefaultSensor()](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#getDefaultSensor(int)) y pasar la constante de tipo para un sensor específico. Si un dispositivo tiene más de un sensor de un tipo determinado, uno de los sensores se debe designar como el sensor predeterminado. Si no existe un sensor predeterminado para un tipo de sensor dado, la llamada al método mostrará un valor nulo, lo que significa que el dispositivo no tiene ese tipo de sensor. Por ejemplo, en el siguiente código, se comprueba si hay un magnetómetro en un dispositivo:

[Kotlin](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#kotlin)

[Java](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#java)

private SensorManager sensorManager;

...

sensorManager = (SensorManager) getSystemService(Context.SENSOR\_SERVICE);

if (sensorManager.getDefaultSensor(Sensor.TYPE\_MAGNETIC\_FIELD) != null){

// Success! There's a magnetometer.

} else {

// Failure! No magnetometer.

}

**Nota:** Android no requiere que los fabricantes de dispositivos construyan ningún tipo particular de sensores en sus dispositivos con Android, por lo que los dispositivos pueden tener una amplia gama de configuraciones de sensor.

Además de enumerar los sensores que están en un dispositivo, puedes usar los métodos públicos de la clase [Sensor](https://developer.android.com/reference/android/hardware/Sensor?hl=es) para determinar las capacidades y los atributos de los sensores individuales. Esto es útil si quieres que tu aplicación se comporte de manera diferente según los sensores o las capacidades de los sensores disponibles en un dispositivo. Por ejemplo, puedes usar los métodos [getResolution()](https://developer.android.com/reference/android/hardware/Sensor?hl=es#getResolution()) y [getMaximumRange()](https://developer.android.com/reference/android/hardware/Sensor?hl=es#getMaximumRange()) para obtener la resolución de un sensor y el rango máximo de medición. También puedes usar el método [getPower()](https://developer.android.com/reference/android/hardware/Sensor?hl=es#getPower()) para obtener los requisitos de energía de un sensor.

Dos de los métodos públicos son particularmente útiles si quieres optimizar tu aplicación para diferentes sensores del fabricante o diferentes versiones de un sensor. Por ejemplo, si tu aplicación necesita supervisar gestos del usuario, como inclinar y agitar, puedes crear un conjunto de reglas y optimizaciones para dispositivos más nuevos, que tengan un sensor de gravedad específico, y otro conjunto de reglas de filtrado de datos y optimizaciones para dispositivos que no tengan sensor de gravedad y solo tengan un acelerómetro. En el siguiente ejemplo de código, se muestra cómo puedes usar los métodos [getVendor()](https://developer.android.com/reference/android/hardware/Sensor?hl=es#getVendor()) y [getVersion()](https://developer.android.com/reference/android/hardware/Sensor?hl=es#getVersion()) para hacer esto. En esta muestra, buscamos un sensor de gravedad que enumere a Google LLC como proveedor y que tenga el número de versión 3. Si ese sensor en particular no está presente en el dispositivo, intentamos usar el acelerómetro.

[Kotlin](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#kotlin)

[Java](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#java)

private SensorManager sensorManager;

private Sensor mSensor;

...

sensorManager = (SensorManager) getSystemService(Context.SENSOR\_SERVICE);

mSensor = null;

if (sensorManager.getDefaultSensor(Sensor.TYPE\_GRAVITY) != null){

List<Sensor> gravSensors = sensorManager.getSensorList(Sensor.TYPE\_GRAVITY);

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

if ((gravSensors.get(i).getVendor().contains("Google LLC")) &&

(gravSensors.get(i).getVersion() == 3)){

// Use the version 3 gravity sensor.

mSensor = gravSensors.get(i);

}

}

}

if (mSensor == null){

// Use the accelerometer.

if (sensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER) != null){

mSensor = sensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER);

} else{

// Sorry, there are no accelerometers on your device.

// You can't play this game.

}

}

Otro método útil es el método [getMinDelay()](https://developer.android.com/reference/android/hardware/Sensor?hl=es#getMinDelay()), que muestra el intervalo de tiempo mínimo (en microsegundos) que un sensor puede usar para detectar datos. Cualquier sensor que muestre un valor distinto de cero para el método [getMinDelay()](https://developer.android.com/reference/android/hardware/Sensor?hl=es#getMinDelay()) es un sensor de transmisión. Los sensores de transmisión detectan datos a intervalos regulares y se introdujeron en Android 2.3 (API nivel 9). Si un sensor muestra cero cuando llamas al método [getMinDelay()](https://developer.android.com/reference/android/hardware/Sensor?hl=es#getMinDelay()), significa que el sensor no es un sensor de transmisión, porque informa datos solo cuando hay un cambio en los parámetros que está detectando.

El método [getMinDelay()](https://developer.android.com/reference/android/hardware/Sensor?hl=es#getMinDelay()) es útil porque te permite determinar la velocidad máxima con la que un sensor puede adquirir datos. Si ciertas características de la aplicación requieren altas velocidades de adquisición de datos o un sensor de transmisión, puedes usar este método para determinar si un sensor cumple con esos requisitos y luego habilitar o inhabilitar las funciones relevantes en tu aplicación según corresponda.

**Precaución:** La velocidad máxima de adquisición de datos de un sensor no es necesariamente la velocidad con la que el marco del sensor entrega los datos a la aplicación. El marco de trabajo del sensor informa los datos a través de eventos del sensor, y varios factores influyen en la velocidad a la que la aplicación recibe los eventos del sensor. Para obtener más información, consulta [Cómo supervisar los eventos del sensor](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#sensors-monitor).

## **Cómo supervisar los eventos del sensor**

Para supervisar los datos sin procesar del sensor, debes implementar dos métodos de devolución de llamada expuestos a través de la interfaz [SensorEventListener](https://developer.android.com/reference/android/hardware/SensorEventListener?hl=es): [onAccuracyChanged()](https://developer.android.com/reference/android/hardware/SensorEventListener?hl=es#onAccuracyChanged(android.hardware.Sensor,%20int)) y [onSensorChanged()](https://developer.android.com/reference/android/hardware/SensorEventListener?hl=es#onSensorChanged(android.hardware.SensorEvent)). El sistema Android llama a estos métodos siempre que ocurre lo siguiente:

* **La exactitud de un sensor cambia.**En este caso, el sistema invoca el método [onAccuracyChanged()](https://developer.android.com/reference/android/hardware/SensorEventListener?hl=es#onAccuracyChanged(android.hardware.Sensor,%20int)), que proporciona una referencia al objeto [Sensor](https://developer.android.com/reference/android/hardware/Sensor?hl=es) que cambió y la nueva exactitud del sensor. La exactitud está representada por una de las cuatro constantes de estado: [SENSOR\_STATUS\_ACCURACY\_LOW](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#SENSOR_STATUS_ACCURACY_LOW), [SENSOR\_STATUS\_ACCURACY\_MEDIUM](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#SENSOR_STATUS_ACCURACY_MEDIUM), [SENSOR\_STATUS\_ACCURACY\_HIGH](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#SENSOR_STATUS_ACCURACY_HIGH) o [SENSOR\_STATUS\_UNRELIABLE](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#SENSOR_STATUS_UNRELIABLE).
* **Un sensor informa un valor nuevo.**En este caso, el sistema invoca el método [onSensorChanged()](https://developer.android.com/reference/android/hardware/SensorEventListener?hl=es#onSensorChanged(android.hardware.SensorEvent)), que proporciona un objeto [SensorEvent](https://developer.android.com/reference/android/hardware/SensorEvent?hl=es). Un objeto [SensorEvent](https://developer.android.com/reference/android/hardware/SensorEvent?hl=es) contiene información sobre los nuevos datos del sensor, por ejemplo, la exactitud de los datos, el sensor que generó los datos, la marca de tiempo en la que se generaron los datos y los nuevos datos que registró el sensor.

En el siguiente código, se muestra cómo usar el método [onSensorChanged()](https://developer.android.com/reference/android/hardware/SensorEventListener?hl=es#onSensorChanged(android.hardware.SensorEvent)) para supervisar los datos del sensor de luz. En este ejemplo, se muestran los datos sin procesar del sensor en un [TextView](https://developer.android.com/reference/android/widget/TextView?hl=es) que se define en el archivo main.xml como sensor\_data.

[Kotlin](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#kotlin)

[Java](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#java)

public class SensorActivity extends Activity implements SensorEventListener {

private SensorManager sensorManager;

private Sensor mLight;

@Override

public final void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

sensorManager = (SensorManager) getSystemService(Context.SENSOR\_SERVICE);

mLight = sensorManager.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();

sensorManager.registerListener(this, mLight, SensorManager.SENSOR\_DELAY\_NORMAL);

}

@Override

protected void onPause() {

super.onPause();

sensorManager.unregisterListener(this);

}

}

En este ejemplo, el retraso de los datos predeterminado ([SENSOR\_DELAY\_NORMAL](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#SENSOR_DELAY_NORMAL)) se especifica cuando se invoca el método [registerListener()](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#registerListener(android.hardware.SensorEventListener,%20android.hardware.Sensor,%20int)). El retraso de los datos (o la tasa de muestreo) controla el intervalo en el que los eventos del sensor se envían a la aplicación a través del método de devolución de llamada [onSensorChanged()](https://developer.android.com/reference/android/hardware/SensorEventListener?hl=es#onSensorChanged(android.hardware.SensorEvent)). El retraso de datos predeterminado es adecuado para supervisar los cambios típicos de orientación de la pantalla y utiliza un retraso de 200,000 microsegundos. Puedes especificar otros retrasos de datos, como [SENSOR\_DELAY\_GAME](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#SENSOR_DELAY_GAME) (retraso de 20,000 microsegundos), [SENSOR\_DELAY\_UI](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#SENSOR_DELAY_UI) (retraso de 60,000 microsegundos) o [SENSOR\_DELAY\_FASTEST](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#SENSOR_DELAY_FASTEST) (retraso de 0 microsegundos). A partir de Android 3.0 (API nivel 11), también puedes especificar el retraso como un valor absoluto (en microsegundos).

El retraso que especificas es solo un retraso sugerido. El sistema Android y otras aplicaciones pueden modificar este retraso. Como práctica recomendada, debes evitar especificar del retraso más largo posible, porque el sistema en general usa un retraso menor que el que especificas (es decir, debes elegir la tasa de muestreo más baja que cumpla con las necesidades de tu aplicación). El uso de un retraso mayor impone una carga menor en el procesador y, por lo tanto, usa menos alimentación.

No existe un método público para determinar la velocidad a la que el marco de trabajo del sensor envía eventos del sensor a tu aplicación; sin embargo, puedes usar las marcas de tiempo asociadas con cada evento del sensor para calcular la tasa de muestreo en varios eventos. No es necesario cambiar la tasa de muestreo (retraso) una vez definida. Si por algún motivo necesitas cambiar el retraso, deberás cancelar el registro y volver a registrar el objeto de escucha.

También es importante tener en cuenta que este ejemplo utiliza los métodos de devolución de llamada [onResume()](https://developer.android.com/reference/android/app/Activity?hl=es#onResume()) y [onPause()](https://developer.android.com/reference/android/app/Activity?hl=es#onPause()) para registrar el evento y cancelar el registro del objeto de escucha del sensor. Como práctica recomendada, siempre debes desactivar los sensores que no necesitas, en especial, cuando la actividad está en pausa. De lo contrario, es posible que la batería se agote en unas horas, ya que algunos sensores tienen requisitos de alimentación intensa y pueden agotar la batería con rapidez. El sistema no inhabilitará los sensores automáticamente cuando la pantalla se apague.

## **Cómo controlar diferentes configuraciones del sensor**

Android no especifica una configuración de sensor estándar para dispositivos, lo que significa que los fabricantes de dispositivos pueden incorporar cualquier configuración de sensor que deseen en sus dispositivos con Android. Como resultado, los dispositivos pueden incluir una variedad de sensores en una amplia gama de configuraciones. Si la aplicación se basa en un tipo específico de sensor, debes asegurarte de que el sensor esté presente en un dispositivo para que la aplicación se pueda ejecutar correctamente.

Tienes dos opciones para garantizar que un sensor determinado esté presente en un dispositivo:

* Detecta sensores en el tiempo de ejecución y habilita o inhabilita las características de la aplicación según corresponda.
* Usa los filtros de Google Play para apuntar a dispositivos con configuraciones de sensor específicas.

En las siguientes secciones, se analiza cada una de las opciones.

#### **Detecta los sensores en el tiempo de ejecución**

Si tu aplicación usa un tipo específico de sensor, pero no depende de él, puedes usar el marco de trabajo del sensor para detectar el sensor en el tiempo de ejecución y luego inhabilitar o habilitar las características de la aplicación según corresponda. Por ejemplo, una aplicación de navegación podría usar el sensor de temperatura, el sensor de presión, el sensor GPS y el sensor del campo geomagnético para mostrar la temperatura, la presión barométrica, la ubicación y la dirección de la brújula. Si un dispositivo no tiene sensor de presión, puedes usar el marco de trabajo del sensor para detectar la ausencia del sensor de presión en el tiempo de ejecución y luego inhabilitar la parte de la IU de la aplicación que muestra la presión. Por ejemplo, en el siguiente código, se comprueba si hay un sensor de presión en un dispositivo:

[Kotlin](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#kotlin)

[Java](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#java)

private SensorManager sensorManager;

...

sensorManager = (SensorManager) getSystemService(Context.SENSOR\_SERVICE);

if (sensorManager.getDefaultSensor(Sensor.TYPE\_PRESSURE) != null){

// Success! There's a pressure sensor.

} else {

// Failure! No pressure sensor.

}

#### **Usa los filtros de Google Play para orientar configuraciones específicas de sensores**

Si estás publicando tu aplicación en Google Play, puedes usar el elemento [<uses-feature>](https://developer.android.com/guide/topics/manifest/uses-feature-element?hl=es) en el archivo de manifiesto para filtrar la aplicación en los dispositivos que no tienen la configuración de sensor adecuada para tu aplicación. El elemento <uses-feature> tiene varios descriptores de hardware que te permiten filtrar aplicaciones según la presencia de sensores específicos. Algunos de los sensores que puedes incluir son los siguientes: acelerómetro, barómetro, brújula (campo geomagnético), giroscopio, luz y proximidad. La siguiente es una entrada de manifiesto de ejemplo que filtra las apps que no tienen un acelerómetro:

<uses-feature android:name="android.hardware.sensor.accelerometer"

android:required="true" />

Si agregas este elemento y descriptor al manifiesto de tu aplicación, los usuarios verán tu aplicación en Google Play solo si el dispositivo tiene un acelerómetro.

Debes establecer el descriptor en android:required="true" solo si tu aplicación se basa completamente en un sensor específico. Si tu aplicación utiliza un sensor para alguna funcionalidad, pero aún se ejecuta sin el sensor, debes incluir el sensor en el elemento <uses-feature>, pero establecer el descriptor en android:required="false". Esto ayuda a garantizar que los dispositivos puedan instalar la app incluso si no tienen ese sensor en particular. Esta es también una práctica recomendada de administración de proyectos que te ayuda a hacer un seguimiento de las funciones que usa tu app. Ten en cuenta que, si la aplicación usa un sensor en particular, pero aún se ejecuta sin el sensor, debes detectar el sensor en el tiempo de ejecución y habilitar o inhabilitar las características de la aplicación según corresponda.

## **Sistema de coordenadas del sensor**

En general, el marco de trabajo del sensor utiliza un sistema de coordenadas estándar de 3 ejes para expresar valores de datos. Para la mayoría de los sensores, el sistema de coordenadas se define en relación con la pantalla del dispositivo cuando el dispositivo se mantiene en su orientación predeterminada (consulta la figura 1). Cuando un dispositivo se mantiene en la orientación predeterminada, el eje X es horizontal y apunta hacia la derecha; el eje Y es vertical y apunta hacia arriba; y el eje Z apunta hacia el exterior de la pantalla. En este sistema, las coordenadas de la pantalla tienen valores Z negativos. Los siguientes sensores utilizan este sistema de coordenadas:



**Figura 1:** Sistema de coordenadas (en relación con un dispositivo) que usa la API del sensor

* [Sensor de aceleración](https://developer.android.com/guide/topics/sensors/sensors_motion?hl=es#sensors-motion-accel)
* [Sensor de gravedad](https://developer.android.com/guide/topics/sensors/sensors_motion?hl=es#sensors-motion-grav)
* [Giroscopio](https://developer.android.com/guide/topics/sensors/sensors_motion?hl=es#sensors-motion-gyro)
* [Sensor de aceleración lineal](https://developer.android.com/guide/topics/sensors/sensors_motion?hl=es#sensors-motion-linear)
* [Sensor de campo geomagnético](https://developer.android.com/guide/topics/sensors/sensors_position?hl=es#sensors-pos-mag)

El punto más importante para comprender este sistema de coordenadas es que los ejes no cambian cuando cambia la orientación de la pantalla; es decir, el sistema de coordenadas nunca cambia a medida que el dispositivo se mueve. Este comportamiento es el mismo que el del sistema de coordenadas OpenGL.

Otro punto que debes comprender es que tu aplicación no debe suponer que la orientación natural (predeterminada) de un dispositivo será la orientación vertical. La orientación natural de muchos dispositivos tablet es horizontal. Además, el sistema de coordenadas del sensor siempre se basa en la orientación natural del dispositivo.

Finalmente, si la aplicación hace coincidir los datos del sensor con la visualización en pantalla, debes usar el método [getRotation()](https://developer.android.com/reference/android/view/Display?hl=es#getRotation()) para determinar la rotación de la pantalla y, luego, usar el método [remapCoordinateSystem()](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#remapCoordinateSystem(float%5B%5D,%20int,%20int,%20float%5B%5D)) para asignar las coordenadas del sensor a las coordenadas de la pantalla. Debes hacer esto incluso si el manifiesto especifica una pantalla de solo retrato.

**Nota:** Algunos sensores y métodos usan un sistema de coordenadas que es relativo al marco de referencia del mundo (en oposición al marco de referencia del dispositivo). Estos sensores y métodos muestran datos que representan el movimiento del dispositivo o la posición del dispositivo en relación con la Tierra. Para obtener más información, consulta el método [**getOrientation()**](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#getOrientation(float%5B%5D,%20float%5B%5D)), el método [**getRotationMatrix()**](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#getRotationMatrix(float%5B%5D,%20float%5B%5D,%20float%5B%5D,%20float%5B%5D)), el [sensor de orientación](https://developer.android.com/guide/topics/sensors/sensors_position?hl=es#sensors-pos-orient) y el [sensor del vector de rotación](https://developer.android.com/guide/topics/sensors/sensors_motion?hl=es#sensors-motion-rotate).

## **Prácticas recomendadas para acceder a los sensores y usarlos**

Cuando diseñes la implementación del sensor, asegúrate de seguir las pautas que se analizan en esta sección. Estas pautas son prácticas recomendadas para cualquier persona que esté usando el marco de trabajo del sensor para acceder a los sensores y adquirir sus datos.

#### **Recopila solo datos de los sensores en segundo plano**

En los dispositivos que ejecutan Android 9 (API nivel 28) o posterior, las aplicaciones que se ejecutan en segundo plano tienen las siguientes restricciones:

* Los sensores que usan el modo de generación de informes [continuo](https://source.android.com/devices/sensors/report-modes?hl=es#continuous), como acelerómetros y giroscopios, no reciben eventos.
* Los sensores que utilizan los modos de generación de informes [cuando se producen cambios](https://source.android.com/devices/sensors/report-modes?hl=es#on-change) o [por única vez](https://source.android.com/devices/sensors/report-modes?hl=es#one-shot) no reciben eventos.

Teniendo en cuenta estas restricciones, es mejor detectar los eventos del sensor cuando la app está en primer plano o como parte de un [servicio en primer plano](https://developer.android.com/guide/components/services?hl=es#Foreground).

#### **Cancela el registro de los objetos de escucha de los sensores**

Asegúrate de cancelar el registro de los objetos de escucha del sensor cuando hayas terminado de usar el sensor o cuando se detenga la actividad del sensor. Si se registra un objeto de escucha del sensor y se detiene la actividad, el sensor continuará adquiriendo datos y consumiendo recursos de la batería, a menos que canceles el registro del sensor. En el siguiente código, se muestra cómo usar el método [onPause()](https://developer.android.com/reference/android/app/Activity?hl=es#onPause()) para cancelar el registro de un objeto de escucha:

[Kotlin](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#kotlin)

[Java](https://developer.android.com/guide/topics/sensors/sensors_overview?hl=es#java)

private SensorManager sensorManager;

...

@Override

protected void onPause() {

super.onPause();

sensorManager.unregisterListener(this);

}

Para obtener más información, consulta [unregisterListener(SensorEventListener)](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#unregisterListener(android.hardware.SensorEventListener)).

#### **Haz una prueba con Android Emulator**

Android Emulator incluye un conjunto de controles de sensores virtuales que permiten probar sensores como el acelerómetro, la temperatura ambiente, el magnetómetro, la proximidad, la luz y más.



El emulador usa una conexión con un dispositivo Android que ejecuta la app [SdkControllerSensor](https://android.googlesource.com/platform/sdk/+/master/apps/SdkController/). Ten en cuenta que esta app solo está disponible en dispositivos con Android 4.0 (API nivel 14) o versiones posteriores. (Si el dispositivo ejecuta Android 4.0, debe tener la revisión 2 instalada). La app *SdkControllerSensor* supervisa los cambios en los sensores del dispositivo y los transmite al emulador. El emulador se transforma en función de los nuevos valores que recibe de los sensores del dispositivo.

Puedes ver el código fuente de la app *SdkControllerSensor* en la siguiente ubicación:

$ **your-android-sdk-directory**/tools/apps/SdkController

Para transferir datos entre tu dispositivo y el emulador, sigue estos pasos:

1. Comprueba que [la depuración de USB esté habilitada](https://developer.android.com/studio/run/device?hl=es#developer-device-options) en tu dispositivo.
2. Conecta el dispositivo a la máquina de desarrollo con un cable USB.
3. Inicia la app *SdkControllerSensor* en tu dispositivo.
4. En la app, selecciona los sensores que quieras emular.
5. Ejecuta el siguiente comando adb:

$ adb forward tcp:1968 tcp:1968

2. Inicia el emulador. Aplica las transformaciones al emulador moviendo el dispositivo.

**Nota:** Si los movimientos que haces en el dispositivo físico no transforman el emulador, intenta ejecutar el comando **adb** del paso 5 de nuevo.

Para obtener más información, consulta la [Guía de Android Emulator](https://developer.android.com/studio/run/emulator?hl=es).

#### **No bloquees el método onSensorChanged()**

Los datos del sensor pueden cambiar a una velocidad alta, lo que significa que es posible que el sistema llame al método [onSensorChanged(SensorEvent)](https://developer.android.com/reference/android/hardware/SensorEventListener?hl=es#onSensorChanged(android.hardware.SensorEvent)) con bastante frecuencia. Como práctica recomendada, debes hacer lo menos posible dentro del método [onSensorChanged(SensorEvent)](https://developer.android.com/reference/android/hardware/SensorEventListener?hl=es#onSensorChanged(android.hardware.SensorEvent)) para no bloquearlo. Si tu aplicación requiere que hagas un filtro de datos o que reduzcas los datos del sensor, debes realizar ese trabajo fuera del método [onSensorChanged(SensorEvent)](https://developer.android.com/reference/android/hardware/SensorEventListener?hl=es#onSensorChanged(android.hardware.SensorEvent)).

#### **Evita usar métodos o tipos de sensores obsoletos**

Varios métodos y constantes ya no están disponibles. En particular, el tipo de sensor [TYPE\_ORIENTATION](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_ORIENTATION) es obsoleto. Para obtener datos de orientación, debes utilizar el método [getOrientation()](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#getOrientation(float%5B%5D,%20float%5B%5D)) en su lugar. Del mismo modo, el tipo de sensor [TYPE\_TEMPERATURE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_TEMPERATURE) es obsoleto. En su lugar, debes usar el tipo de sensor [TYPE\_AMBIENT\_TEMPERATURE](https://developer.android.com/reference/android/hardware/Sensor?hl=es#TYPE_AMBIENT_TEMPERATURE) en dispositivos que ejecutan Android 4.0.

#### **Comprueba los sensores antes de usarlos**

Siempre comprueba que existe un sensor en un dispositivo antes de intentar obtener datos de él. No supongas que existe un sensor simplemente porque es un sensor de uso frecuente. Los fabricantes de dispositivos no están obligados a proporcionar sensores particulares en sus dispositivos.

#### **Elige los retrasos del sensor con cuidado**

Cuando registres un sensor con el método [registerListener()](https://developer.android.com/reference/android/hardware/SensorManager?hl=es#registerListener(android.hardware.SensorEventListener,%20android.hardware.Sensor,%20int)), asegúrate de elegir un ritmo de entrega adecuado para tu aplicación o caso de uso. Los sensores pueden proporcionar datos a ritmos muy rápidos. Si permites que el sistema envíe datos adicionales que no necesitas, se desperdician recursos del sistema y se consume batería.