



1-Definicion

2-Categorias y tipos

3-Utilizacion en Android

4-Datos de interés

5-Ejemplos de aplicaciones

6-Bibliografia

¿Qué son?

Un conjunto de dispositivos con los que podremos obtener información del mundo exterior (en este conjunto no se incluye la cámara, el micrófono o el GPS).

Son los dispositivos de entrada más novedosos que incorpora Android y con ellos podremos implementar formas atractivas de interacción con el usuario.

*Los sensores se dividen en tres categorías:

→ Sensores de movimiento en nuestro Android

Estos sensores miden las fuerzas de aceleración y fuerzas de rotación a lo largo de tres ejes. En esta categoría incluimos los acelerómetros, sensores de gravedad, giroscopios y sensores de rotación del vector.

→ Sensores ambientales en nuestro Android

Estos sensores miden diversos parámetros ambientales, como la temperatura del aire ambiente y la presión, la iluminación, y la humedad. En Esta categoría incluimos: barómetros, fotómetros y termómetros.

→ Sensores de posición en nuestro Android

Estos sensores miden la posición física de un dispositivo. Esta categoría incluye los sensores de orientación y magnetómetros.

*Tipos de Sensores soportados en la plataforma Android.

Sensor	Type	Description	Common Uses
TYPE_ACCELEROMETER	Hardware	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_AMBIENT_TEMPERATURE	Hardware	Measures the ambient room temperature in degrees Celsius ($^{\circ}C$). See note below.	Monitoring air temperatures.
TYPE_GRAVITY	Software or Hardware	Measures the force of gravity in m/s^2 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/s^2 that is applied to a device on all three physical axes (x, y, and z).	Monitoring acceleration along a single

		excluding the force of gravity.	axis.
TYPE_MAGNETIC_FIELD	Hardware	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in μT .	Creating a compass.
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 <code>getRotationMatrix()</code> 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 ($^{\circ}\text{C}$). This sensor implementation varies across devices and this sensor was replaced with the <code>TYPE_AMBIENT_TEMPERATURE</code> sensor in API Level 14	Monitoring temperatures.

*Sensores disponibles por plataforma.

Sensor	Android 4.0 (API Level 14)	Android 2.3 (API Level 9)	Android 2.2 (API Level 8)	Android 1.5 (API Level 3)
TYPE_ACCELEROMETER	Yes	Yes	Yes	Yes
TYPE_AMBIENT_TEMPERATURE	Yes	n/a	n/a	n/a
TYPE_GRAVITY	Yes	Yes	n/a	n/a
TYPE_GYROSCOPE	Yes	Yes	n/a ¹	n/a ¹
TYPE_LIGHT	Yes	Yes	Yes	Yes
TYPE_LINEAR_ACCELERATION	Yes	Yes	n/a	n/a
TYPE_MAGNETIC_FIELD	Yes	Yes	Yes	Yes
TYPE_ORIENTATION	Yes ²	Yes ²	Yes ²	Yes
TYPE_PRESSURE	Yes	Yes	n/a ¹	n/a ¹
TYPE_PROXIMITY	Yes	Yes	Yes	Yes
TYPE_RELATIVE_HUMIDITY	Yes	n/a	n/a	n/a

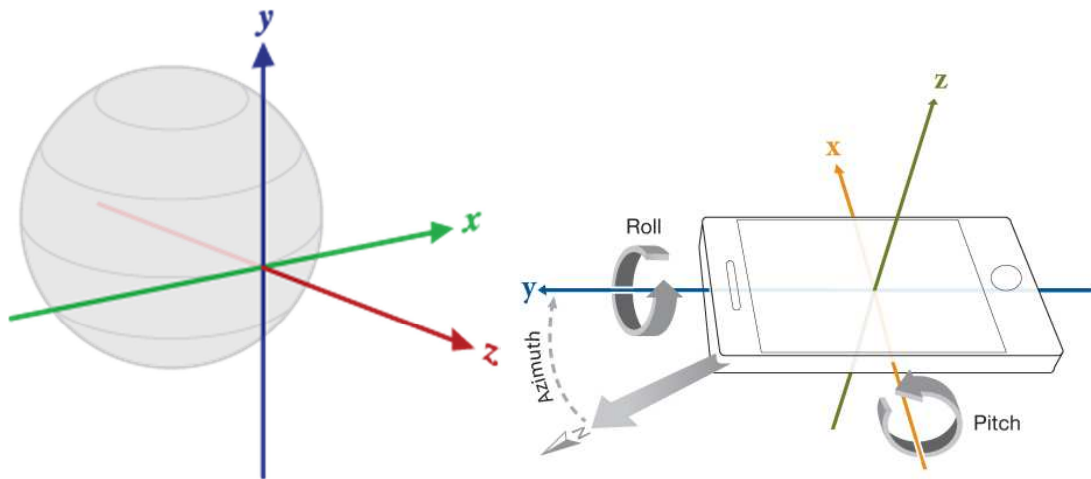
TYPE_ROTATION_VECTOR	Yes	Yes	n/a	n/a
TYPE_TEMPERATURE	Yes ²	Yes	Yes	Yes

1-Sensores de movimientos en Android

En esta categoría incluimos los acelerómetros, sensores de gravedad, giroscopios y sensores de rotación del vector.

La plataforma Android ofrece varios sensores que le permiten controlar el movimiento de un dispositivo. Dos de estos sensores son siempre basado en hardware (el acelerómetro y un giroscopio), y tres de estos sensores puede ser o bien (sensores de rotación del vector de la gravedad, la aceleración lineal, y) basada en hardware o basada en software

Todos los sensores de movimiento devuelven matrices multidimensionales de valores de los sensores para cada SensorEvent .



Sensor	Datos de eventos del sensor	Descripción	Las unidades de medida
TYPE_ACCELEROMETER	<code>SensorEvent.values [0]</code>	Fuerza de aceleración a lo largo del eje X (incluyendo la gravedad).	m / s^2
	<code>SensorEvent.values [1]</code>	Fuerza de aceleración a lo largo del eje Y (incluyendo la gravedad).	
	<code>SensorEvent.values [2]</code>	Fuerza de aceleración a lo largo del eje z (incluyendo la gravedad).	
TYPE_GYROSCOPE	<code>SensorEvent.values [0]</code>	Tasa de rotación alrededor del eje x.	rad / s
	<code>SensorEvent.values [1]</code>	Tasa de rotación alrededor del eje y.	
	<code>SensorEvent.values [2]</code>	Tasa de rotación alrededor del eje z.	
TYPE_LINEAR_ACCELERATION	<code>SensorEvent.values [0]</code>	Fuerza de aceleración a lo largo del eje x (con exclusión de la gravedad).	m / s^2
	<code>SensorEvent.values [1]</code>	Fuerza de aceleración a lo largo del eje Y (con exclusión de la gravedad).	
	<code>SensorEvent.values [2]</code>	Fuerza de aceleración a lo largo del eje z (con exclusión de la gravedad).	

TYPE_ROTATION_VECTOR	SensorEvent.values [0]	Rotación componente de vector a lo largo del eje x ($x * \sin(\theta / 2)$).	Sin unidad
	SensorEvent.values [1]	Componente de vector de rotación a lo largo del eje y ($y * \sin(\theta / 2)$).	
	SensorEvent.values [2]	Rotación componente de vector a lo largo del eje z ($z * \sin(\theta / 2)$).	
	SensorEvent.values [3]	Componente escalar del vector de rotación ($\cos(\theta / 2)$).	

2-Sensores de posición

Dos sensores que permiten determinar la posición de un dispositivo: el sensor de campo magnético terrestre y el sensor de orientación.

Otro sensor que permite determinar la cercanía hacia el dispositivo es el de proximidad

Los sensores de posición son útiles para determinar la posición física de un dispositivo, se puede utilizar el sensor de campo magnético terrestre en combinación con el acelerómetro para determinar la posición de un dispositivo en relación con el Polo Norte magnético. Ej brújula

Sensor	Datos de eventos del sensor	Descripción	Las unidades de medida
TYPE_MAGNETIC_FIELD	SensorEvent.values [0]	Intensidad de campo geomagnético a lo largo del eje x.	mT
	SensorEvent.values [1]	Intensidad de campo geomagnético a lo largo del eje y.	
	SensorEvent.values [2]	Intensidad de campo geomagnético a lo largo del eje z.	
TYPE_ORIENTATION ¹	SensorEvent.values [0]	Azimut (ángulo alrededor del eje z).	Grados
	SensorEvent.values [1]	Paso (ángulo alrededor del eje x).	
	SensorEvent.values [2]	Rollo (ángulo alrededor del eje y).	
TYPE_PROXIMITY	SensorEvent.values [0]	Distancia del objeto. ²	cm

3-Sensores medio ambiente

Cuatro sensores que le permiten supervisar diversas propiedades ambientales.

Su uso: controlar la humedad relativa del ambiente, la iluminación, la presión ambiente, y la temperatura ambiente cerca de un dispositivo con Android. Los cuatro sensores de entorno están basados en hardware y sólo están disponibles si un fabricante de dispositivos los ha integrado en un dispositivo, con la excepción del sensor de luz, lo que la mayoría de los fabricantes de dispositivos utilizan para controlar el brillo de la pantalla.

Sensor	Sensor event data	Units of measure	Data description
TYPE_AMBIENT_TEMPERATURE	event.values[0]	°C	Ambient air temperature.
TYPE_LIGHT	event.values[0]	lx	Illuminance.
TYPE_PRESSURE	event.values[0]	hPa or mbar	Ambient air pressure.
TYPE_RELATIVE_HUMIDITY	event.values[0]	%	Ambient relative humidity.
TYPE_TEMPERATURE	event.values[0]	°C	Device temperature. ¹

Utilizacion de un sensor en Android

Para el manejo de los diferentes sensores disponibles es necesario hacer uso de estas clases:

[SensorManager](#) → que nos permite acceder a los sensores del dispositivo y la Interfaz.

Se usa para crear una instancia del servicio del sensor. Esta clase proporciona varios métodos para acceder y enumerar los sensores, registrar y anular el registro de los detectores de eventos del sensor, y la adquisición de información de orientación. Esta clase también proporciona varias constantes de sensores que se utilizan para informar de la precisión del sensor, establece las velocidades de adquisición de datos, y calibrar sensores.

[Sensor](#) → representa al sensor que queremos utilizar.

Crear una instancia de un sensor específico. Esta clase proporciona varios métodos que le permiten determinar las capacidades de un sensor.

[SensorEvent](#) → informacion

El sistema utiliza esta clase para crear un objeto de evento del sensor, que proporciona información acerca de un evento de sensor. Un objeto de evento del sensor incluye la siguiente información: los datos de los sensores primas, el tipo de sensor que generó el evento, la exactitud de los datos, y la marca de tiempo para el evento.

[SensorEventListener](#) → que registra los cambios hechos en el sensor indicado mediante una variación de precision.

Paso 1-Crear la Actividad

La actividad que vamos a crear hereda de *Activity* para otorgarle las características de una actividad e implementa *SensorEventListener* para el registro de cambios en el sensor

```
public class SensorActivity extends Activity implements SensorEventListener {  
  
}
```

Paso 2- definir las variables a utilizar

Variables del tipo *Sensor* los cuales utilizaré para guardar una instancia de cada sensor que pueda detectar usando un objeto de [SensorManager](#)

```
private SensorManager mSensorManager;  
private Sensor sensor;
```

Paso 3- Instanciar sensores

Para poder obtener instancias de los sensores disponibles en el dispositivo Android, se utiliza [SensorManager](#) a través de la llamada al método [getSystemService\(\)](#) que nos retorna un servicio a nivel de sistema dependiendo del parámetro que le pasemos, en este caso *SENSOR_SERVICE*, pues queremos hacer uso de los sensores.

```
mSensorManager = SensorManager.getSystemService(Context.SENSOR_SERVICE);
```

Una vez inicializado [SensorManager](#), podemos hacer uso de este objeto para solicitar instancias de los diferentes tipo de sensores haciendo uso del metodo [getDefaultSensor\(\)](#) y añadiendo el tipo de sensor que queremos como parametro

```
sensor = mSensorManager.getDefaultSensor(Sensor.TYPE);
```

TYPE_ACCELEROMETER-acelerometro

TYPE_ORIENTATION-orientacion

TYPE_MAGNETIC_FIELD -brujula(campo magnetico)

TYPE_PROXIMITY-proximidad

TYPE_LIGHT-luz

TYPE_TEMPERATURE-temperatura

TYPE_GRAVITY -gravedad

TYPE_ROTATION_VECTOR-vector de rotacion

TYPE_PRESSURE-presion

Paso 4-

Implementación de los métodos de la Interfaz

La Interfaz [SensorEventListener](#) nos pide implementar 2 métodos:

- *onAccuracyChanged(Sensor sensor, int accuracy)*, en la cual implementaremos las acciones a realizar cuando se cambia la precision de un sensor.

- *onSensorChanged(SensorEvent event)*, la cual nos permite implementar las acciones a realizar cuando un sensor registre un cambio.

Codigo de ejemplo para saber el sensor de proximidad

@Override

```
public void onSensorChanged(SensorEvent event)
{
    if(event.sensor.getType() == Sensor.TYPE_PROXIMITY)
    {
        Toast.makeText(getApplicationContext(), "Existe el sensor de proximidad",
        Toast.LENGTH_SHORT).show();
    }
}
```

IMPORTANTE*****

El uso de los sensores requiere de más energía por parte de la aplicación a diferencia de las que no lo hacen, por lo que es recomendable liberar al manejador de eventos cuando se vaya a salir de la aplicación e implementarla de nuevo cuando se vaya a hacer uso de ésta nuevamente. Esto se puede hacer de la siguiente manera:

```
public void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, sensor, SensorManager.SENSOR_DELAY_NORMAL);
}

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

4-Datos de interés

*Clase Paint

-Esta clase representa un pincel. nos permite definir el color, estilo o grosor del trazado de un gráfico vectorial.

Metodos importantes:

*Color del pincel → setColor(int color)

setAlpha(int alfa), para el grado de transparencia

*Tipo de trazado → setStrokeWidth(float grosor) define el grosor del trazado

```
setStyle( Paint.Style estilo) *valores FILL,FILL_AND_STROKE,
STROKE
```

```
setShadowLayer(float radio, float dx, float dy, int color); realizar un
segundo trazado en forma de sombra.
```

*Tipo de texto → setTextAlign(Paint.Align justif) Justificar el texto CENTER,LEFT,RIGH

setTextSize(float tamaño) tamaño del texto

.....

setTypeface(Typeface fuente) MONOSPACE,SERIF,SANS_SERIF

.....

Asociada a la clase paint, esta la clase canvas

La clase canvas representa una superficie donde podemos dibujar, es decir el lugar donde se pueden representar figuras geométricas, textos, fotos, etc.

-Para dibujar figuras geométricas

-Para dibujar líneas y arcos

-Para dibujar texto

-Para rellenar todo el Canvas

-Para dibujar imágenes

.....

código de la actividad sería:

```
public class EjemploGraficosActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new EjemploView(this));
    }

    public class EjemploView extends View {
        public EjemploView (Context context) {
            super(context);
        }
        @Override
        protected void onDraw(Canvas canvas) {
            //Dibujar aquí
        }
    }
}
```

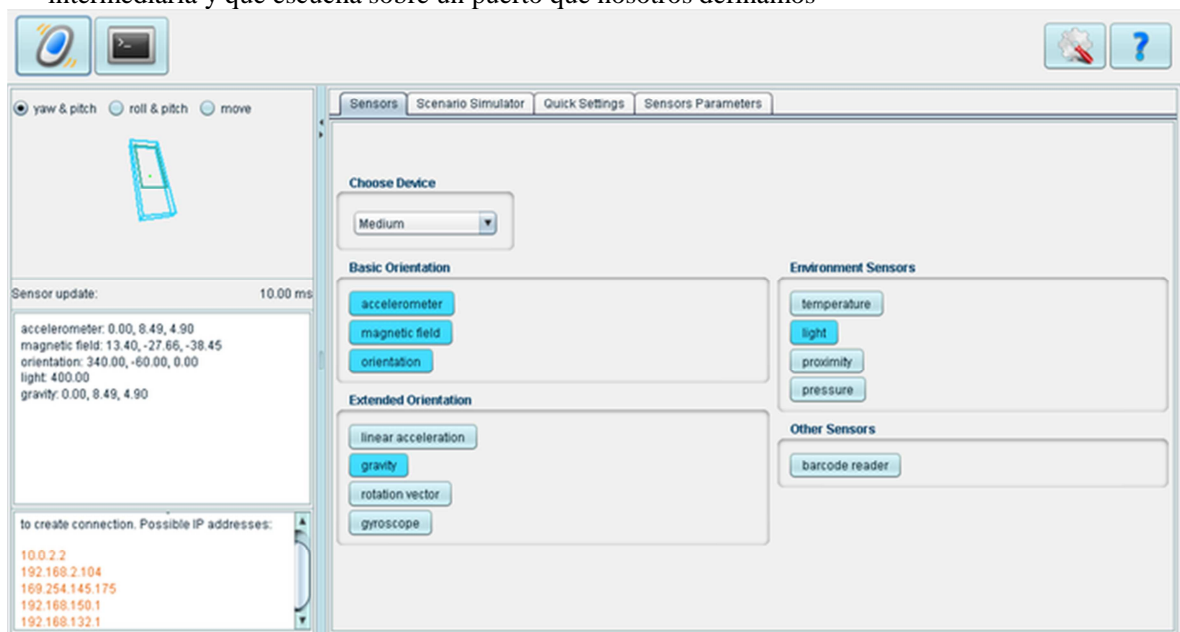
Comienza con la creación de una **Activity**, pero en este caso, el objeto **View** que asociamos a la actividad mediante el método **setContentView()** no está definido mediante XML. Por el contrario, es creado mediante código a partir del constructor de la clase **EjemploView**.

La clase **EjemploView** extiende **View**, modificando solo el método **onDraw()** responsable de dibujar la vista.

*Sensor Simulator

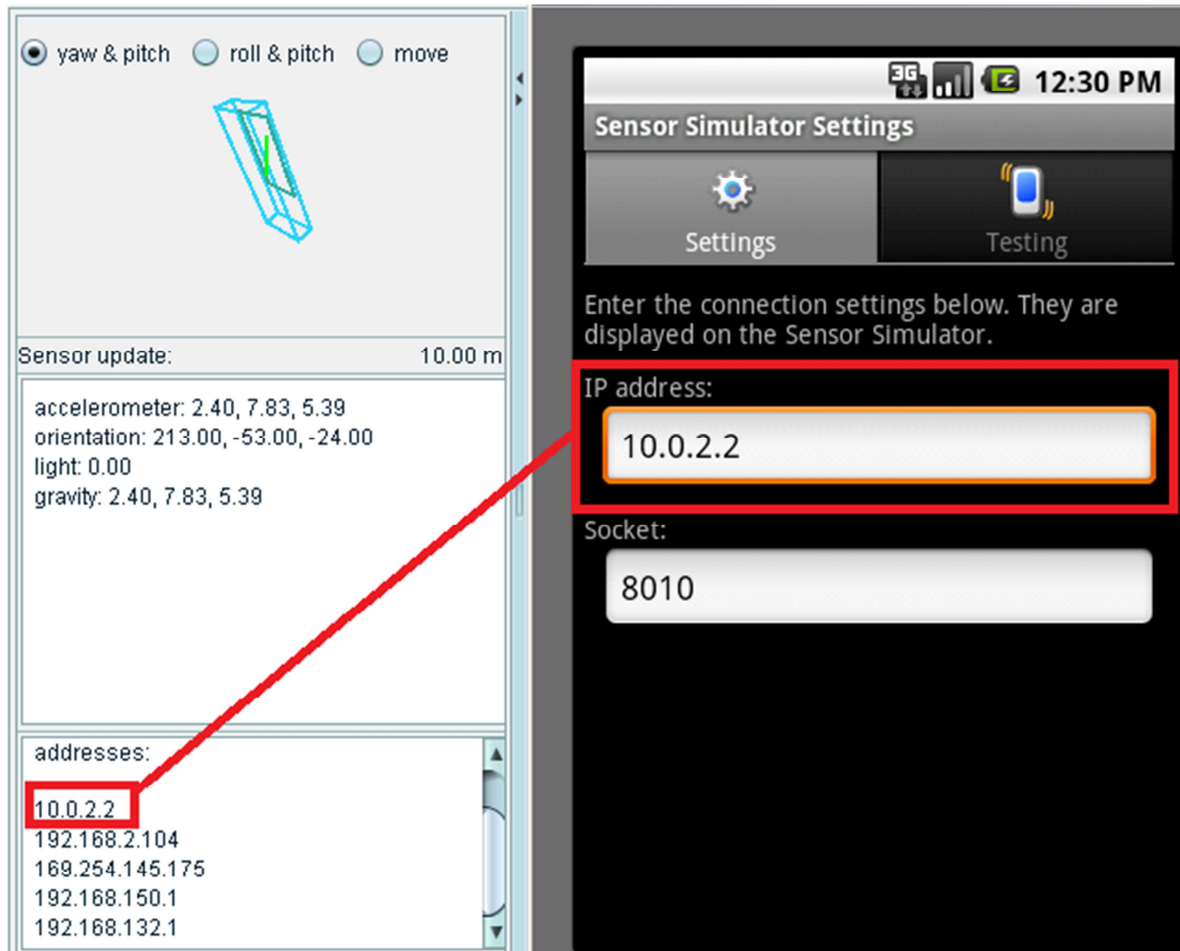
Sensor simulator es una aplicación independiente de Java que simula los datos de los sensores y los transmite al emulador de Android. Actualmente el simulador cuenta con las siguientes funciones:

- Simula los sensores: de orientación, acelerómetro y brújula, así como un sensor de temperatura.
- El dispositivo móvil puede ser manipulado a través del ratón o los controles deslizantes, simulando así las distintas posiciones y orientaciones de un dispositivo real.
- Es compatible con la API de sensores de Android.
- Realiza la conexión con el emulador de Android a través de una aplicación que funciona como intermediaria y que escucha sobre un puerto que nosotros definamos

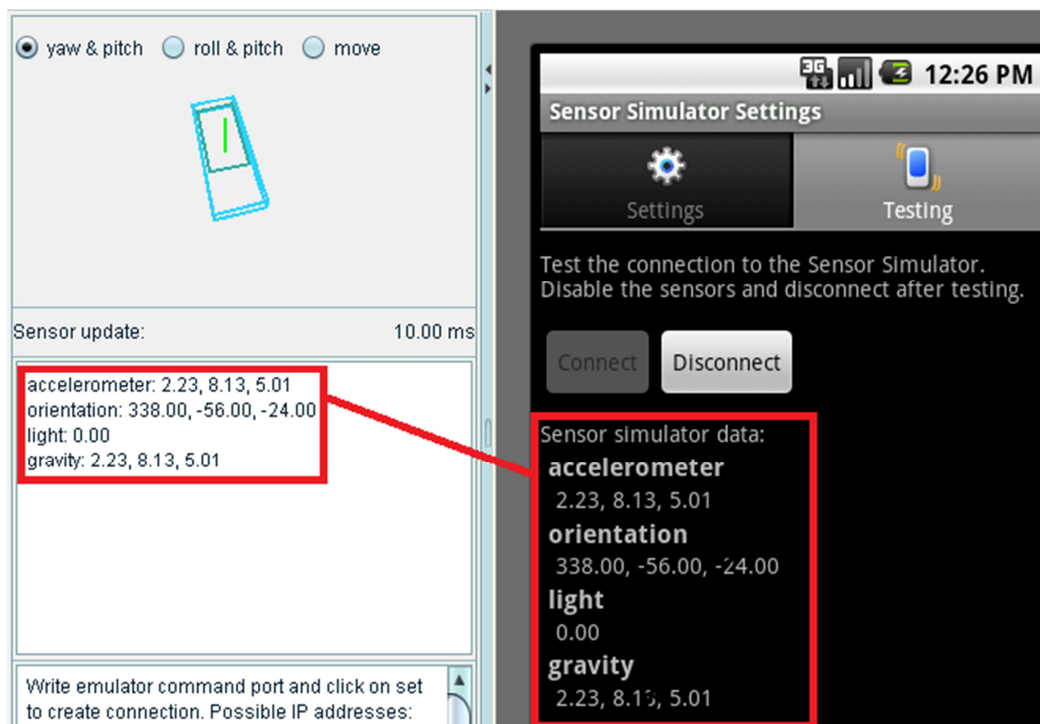


¿Como ejecutar nuestra aplicación con el simulador?

- Descargar el programa
- Ejecutar el jar sensorsimulator-x.x.x.jar que está en la carpeta bin (x significa número de versión o estado del paquete).
- Instalar el bin/SensorSimulatorSettings.apk en el emulador de Android.
 - Con el emulador de Android corriendo, ejecutar desde línea de comandos y posicionado en la carpeta tools de la instalación del SDK de Android: **adb install /path/al/apk.apk**
- Ejecutar la aplicación SensorSimulatorSettings en el emulador de Android.
- Ingresar la dirección IP y el número de socket (port) que tiene el Simulador de sensores



Se debe conectar la aplicación Sensor Simulator Settings desde el tab Testing con el SensorSimulator



¿Como usar el simulador en nuestra aplicación?

Incluir el JAR "lib/sensorsimulator-lib.jar" en el proyecto. Importar las clases del SensorSimulator

```
import org.openintents.sensorsimulator.hardware.Sensor;

import org.openintents.sensorsimulator.hardware.SensorEvent;

import org.openintents.sensorsimulator.hardware.SensorEventListener;

import org.openintents.sensorsimulator.hardware.SensorManagerSimulator;
```

Reemplazar el siguiente código en el onCreate():

```
mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
```

por este otro

```
mSensorManager = SensorManagerSimulator.getSystemService(this,
SENSOR_SERVICE);
```

Conectar con el sensor simulator utilizando:

```
mSensorManager.connectSimulator();
```

5-EJEMPLOS APLICACIONES EN ANDROID

Ejemplo sencillo-v1

```
package com.mvs.ejemplosencillo;
```

```
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.TextView;
```

```
public class MainActivity extends Activity implements SensorEventListener {
```

```
private SensorManager mSensorManager ;
private Sensor sensor;

private TextView datosSensor;

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

    mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    sensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

    datosSensor = (TextView) findViewById(R.id.textView1);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override
public void onAccuracyChanged(Sensor arg0, int arg1) {
    // TODO Auto-generated method stub
}

@Override
public void onSensorChanged(SensorEvent event) {
    // TODO Auto-generated method stub

    String datos = "DATOS SENSOR \nx: " + event.values[0]+"\n y: " + event.values[1]
    +"\n z: " + event.values[2];

    datosSensor.setText(datos);
}

public void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, sensor, SensorManager.SENSOR_DELAY_NORMAL);
}

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




Ejemplo sencillo v2

```
package com.mvs.ejemplosencillo_v2;

import java.util.List;

import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.app.Activity;
import android.widget.TextView;

public class MainActivity extends Activity {

    private TextView sensores;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensores = (TextView) findViewById(R.id.sensores);
        SensorManager sensorManager = (SensorManager)
        getSystemService(SENSOR_SERVICE);


        List<Sensor> listaSensores = sensorManager
            .getSensorList(Sensor.TYPE_ALL);
        for (Sensor sensor : listaSensores) {
            log(sensor.getName());
        }
    }

    private void log(String string) {
```

```
        sensores.append(string + "\n");  
    }  
}
```



Sensores 2.0 → App que te lista todos los sensores disponibles en tu móvil, y una prueba con cada uno de ellos.

 Sensores en mi Movil

Que sensores tengo?

Limpia

Proximidad Detectada

Vibracion Detectada

Sensor: ACELEROMETRO

x: -0,572 m/s - Max: 14,642
y: 7,641 m/s - Max: 9,439
z: 6,456 m/s - Max: 16,14

Sensor: GRAVEDAD

x: -0.6164031
y: 7.51973
z: 6.2645106

Sensor: VECTOR ROTACION

x: 0.06929606
y: -0.41928142
z: -0.9021659

Pos: Vertical

display: 0

Sensor: ORIENTACION

azimut: S
y: -49.0708¼
z: -6.4483733¼

Sensor: CAMPO MAGNETICO

-1.0799999 uT

Sensor: PROXIMIDAD

5.0

K3DH Acceleration Sensor

AK8975 Magnetic field Sensor

CM3663 Light Sensor

CM3663 Proximity Sensor

K3G Gyroscope Sensor

Rotation Vector Sensor

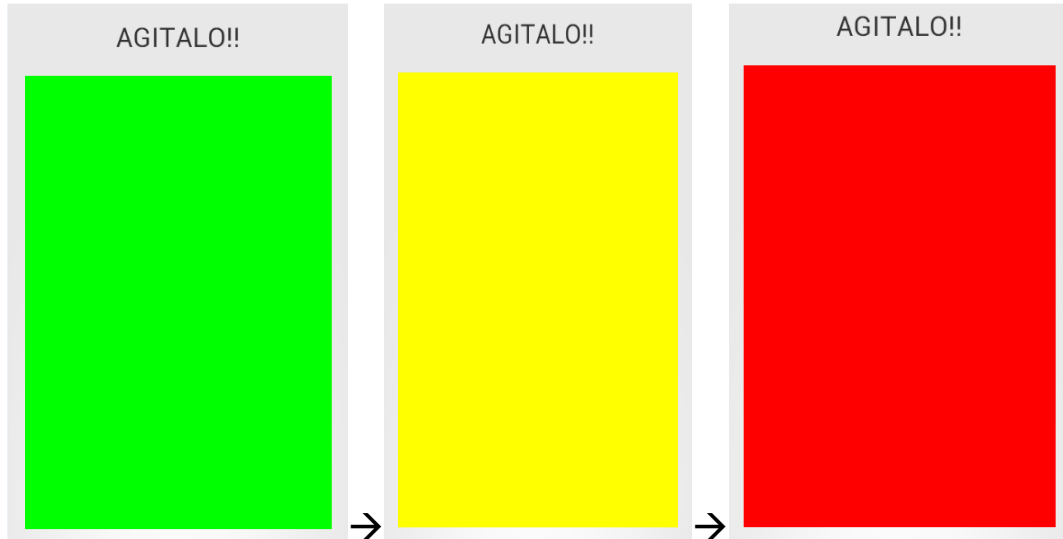
Gravity Sensor

Linear Acceleration Sensor

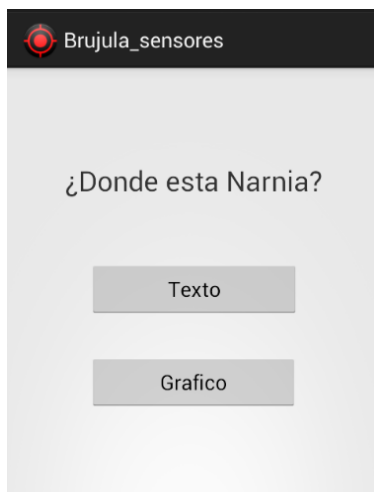
Orientation Sensor

Otras aplicaciones de sensores bastante aparentes:

-Agitar sensores: Si nuestro móvil percibe una vibración, el textView cambiara de color. Por ejemplo:

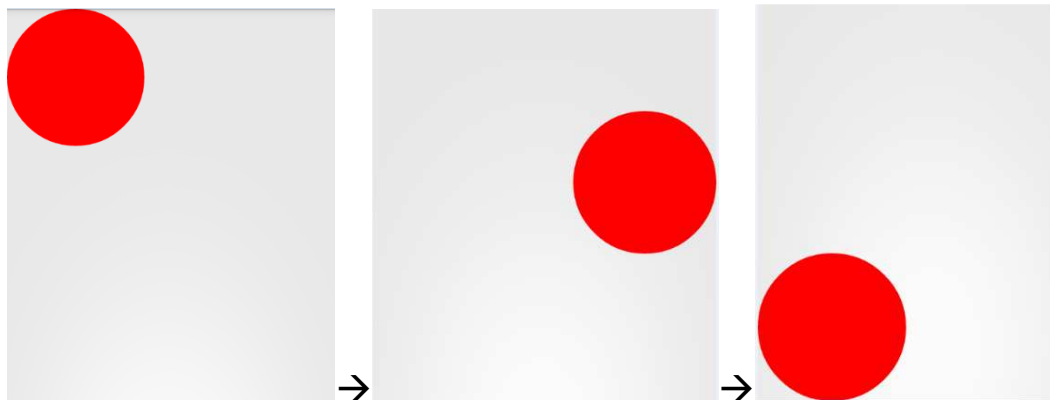


-Brujula sensores: gracias al sensor de orientación y, a la utilización de la clase Paint en el modo grafico, nos orientara hacia donde está el norte magnético.





-Pelota_sensores: Utilizando el sensor acelerómetro, va a tomar sus valores x-y-z para representar una pelota con Paint y canvas y moverla en la pantalla.



6-Bibliografia

1-Sensores:

http://www.aprendeandroid.com/19/sensores_android.htm

<http://stackoverflow.com/search?q=sensors>

http://developer.android.com/guide/topics/sensors/sensors_overview.html

http://developer.android.com/guide/topics/sensors/sensors_motion.html

<http://androideity.com/2011/10/27/deteccion-y-manejo-de-sensores/>

<http://www.vogella.com/tutorials/AndroidSensor/article.html>

http://www.youtube.com/watch?v=_983ZZwCtgw&list=PL7EA29F3B739286CA&index=7 → a partir del minuto 15

2-La clase Paint de Android:

<http://developer.android.com/reference/android/graphics/Paint.html>

<https://polimedia.upv.es/visor/?id=0a62eba0-1e44-f843-9943-e7be0c09712c#>

3-AndoridSimulator:

<http://developer.samsung.com/android/tools-sdks/Samsung-Sensor-Simulator>

<http://www.botskool.com/geeks/how-use-sensor-simulator-android-sdk-emulator>