# 传感器

## 传感器的分类

方向传感器

陀螺仪传感器

磁场传感器

重力传感器

线性加速度传感器

温度传感器

光传感器

压力传感器

心率传感器

实例获取各传感器数据并展示

## 传感器的介绍

Android系统提供了对传感器的支持，如果手机的硬件提供了这些传感器的话，那么我们就可以通过代码获取手机外部的状态。比如说手机的摆放状态、外界的磁场、温度和压力等等。

传感器开发

对于我们开发者来说，开发传感器十分简单。只需要注册监听器，接收回调的数据就行了，下面来详细介绍下各传感器的开发。

### 使用

第一步

// 获取传感器管理对象

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

第二步

// 获取传感器的类型(TYPE\_ACCELEROMETER:加速度传感器)

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

这里我们除了可以获取加速度传感器之外，还可以获取其他类型的传感器，如：

\* Sensor.TYPE\_ORIENTATION：方向传感器。

\* Sensor.TYPE\_GYROSCOPE：陀螺仪传感器。

\* Sensor.TYPE\_MAGNETIC\_FIELD：磁场传感器。

\* Sensor.TYPE\_GRAVITY：重力传感器。

\* Sensor.TYPE\_LINEAR\_ACCELERATION：线性加速度传感器。

\* Sensor.TYPE\_AMBIENT\_TEMPERATURE：温度传感器。

\* Sensor.TYPE\_LIGHT：光传感器。

\* Sensor.TYPE\_PRESSURE：压力传感器。

第三步

在onResume()方法中监听传感器传回的数据：

@Override

protected void onResume() {

super.onResume();

// 为加速度传感器注册监听器

mSensorManager.registerListener(new SensorEventListener() {

// 当传感器的值改变的时候回调该方法

@Override

public void onSensorChanged(SensorEvent event) {

}

// 当传感器精度发生改变时回调该方法

@Override

public void onAccuracyChanged(Sensor sensor, int accuracy) {

}

}, mSensor, SensorManager.SENSOR\_DELAY\_GAME);

}

其中，registerListener(SensorEventListener listener, Sensor sensor,int samplingPeriodUs)的三个参数说明如下：

listener：监听传感器时间的监听器，该监听器需要实现SensorEventListener接口。

sensor：传感器对象。

samplingPeriodUs：指定获取传感器频率，一共有如下几种：

\* SensorManager.SENSOR\_DELAY\_FASTEST：最快，延迟最小，同时也最消耗资源，一般只有特别依赖传感器的应用使用该频率，否则不推荐。

\* SensorManager.SENSOR\_DELAY\_GAME：适合游戏的频率，一般有实时性要求的应用适合使用这种频率。

\* SensorManager.SENSOR\_DELAY\_NORMAL：正常频率，一般对实时性要求不高的应用适合使用这种频率。

\* SensorManager.SENSOR\_DELAY\_UI：适合普通应用的频率，这种模式比较省电，而且系统开销小，但延迟大，因此只适合普通小程序使用。

并在onStop()方法中取消注册：

@Override

protected void onStop() {

super.onStop();

// 取消监听

mSensorManager.unregisterListener(this);

}

简单3步，就完成了监听加速度传感器的开发，是不是so easy？

### 列子

下面一个列子，演示了完整的监听加速度传感器的开发，并将结果显示到屏幕上：

public class MainActivity extends AppCompatActivity implements SensorEventListener{

private SensorManager mSensorManager;

private TextView mTxtValue;

private Sensor mSensor;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mTxtValue = (TextView) findViewById(R.id.txt\_value);

// 获取传感器管理对象

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

// 获取传感器的类型(TYPE\_ACCELEROMETER:加速度传感器)

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

}

@Override

protected void onResume() {

super.onResume();

// 为加速度传感器注册监听器

mSensorManager.registerListener(this, mSensor, SensorManager.SENSOR\_DELAY\_GAME);

}

@Override

protected void onStop() {

super.onStop();

// 取消监听

mSensorManager.unregisterListener(this);

}

// 当传感器的值改变的时候回调该方法

@Override

public void onSensorChanged(SensorEvent event) {

float[] values = event.values;

StringBuilder sb = new StringBuilder();

sb.append("X方向的加速度：");

sb.append(values[0]);

sb.append("\nY方向的加速度：");

sb.append(values[1]);

sb.append("\nZ方向的加速度：");

sb.append(values[2]);

mTxtValue.setText(sb.toString());

}

// 当传感器精度发生改变时回调该方法

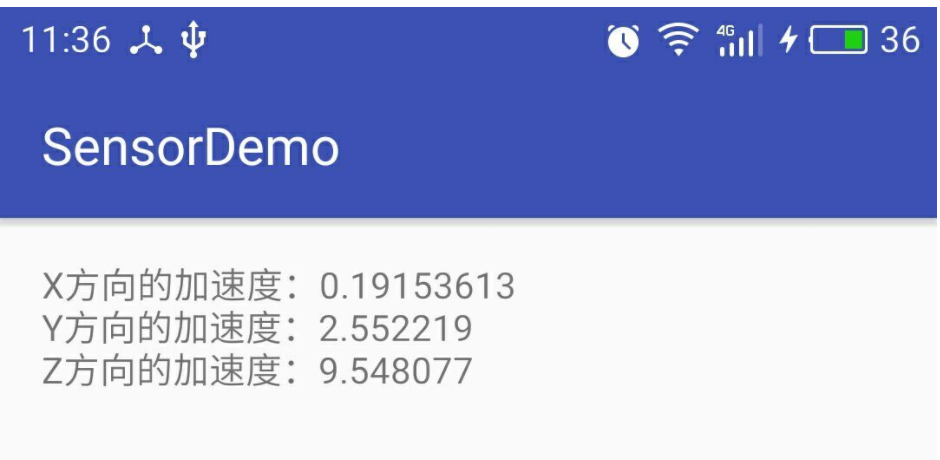
@Override

public void onAccuracyChanged(Sensor sensor, int accuracy) {

}

}

运行结果：



方向传感器(高版本后废弃)

方向传感器用于感应手机的摆放位置，它给我们返回了三个角度，这三个角度可以确定手机的摆放状态。

\* 第一个角度：表示手机顶部朝向与正北方的夹角。当手机绕着Z轴旋转时，该角度值发生改变。比如，当该角度为0度时，表明手机顶部朝向正北；该角度为90度时，表明手机顶部朝向正东；该角度为180度时，表明手机朝向正南；该角度为270度时，表明手机顶部朝向正西。

\* 第二个角度：表示手机顶部或尾部翘起的高度。当手机绕着X轴倾斜时，该角度值发生变化，该角度的取值范围是-180~180度。假设手机屏幕朝上水平放在桌子上，如果桌子是完全水平的，该角度值应该是0度。假如从手机顶部开始抬起，直到将手机沿X轴旋转180度(屏幕向下水平放在桌子上)，在这个旋转的过程中，该角度值会从0度变化到-180度。也就是说，从手机顶部抬起时，该角度的值会逐渐减少，直到等于-180度；如果从手机底部开始抬起，直到将手机沿X轴旋转180度(屏幕向下水平放在桌子上)，该角度的值会从0度变化到180度，也就是说，从手机底部抬起时，该角度的值会逐渐增大，直到等于180度。

\* 第三个角度：表示手机左侧或右侧翘起的角度。当手机绕着Y轴倾斜时，该角度值发生改变。该角度的取值范围是：-90~90度。假设将手机屏幕朝上水平放在桌面上，如果桌面是完全水平的，该角度应该为0度。如果将手机从左侧开始慢慢抬起，知道将手机沿着Y轴旋转90度(手机与桌面垂直)，在这个旋转的过程中，该角度值会从0度变化到-90度。也就是说，从手机左侧开始抬起时，该角度的值会逐渐的减少，知道等于-90度。如果从手机的右侧抬起，则刚好相反，会从0度变化，直到90度。

通过在应用程序中使用方向传感器，可以实现如：地图导航、水平仪、指南针等应用。

陀螺仪传感器

陀螺仪传感器用于感应手机的旋转速度。陀螺仪传感器给我们返回了当前设备的X、Y、Z三个坐标轴（坐标系统与加速度传感器一模一样）的旋转速度。旋转速度的单位是弧度/秒，旋转速度为：

正值代表逆时针旋转，负值代表顺时针旋转。关于返回的三个角速度说明如下：

\* 第一个值：代表该设备绕X轴旋转的角速度。

\* 第二个值：代表该设备绕Y轴旋转的角速度。

\* 第三个值：代表该设备绕Z轴旋转的角速度。

磁场传感器

磁场感应器主要读取设备周围的磁场强度。即便是设备周围没有任何直接的磁场，设备也会始终处于地球的磁场中，除非你不在地球。。随着手机设备摆放状态的改变，周围磁场在手机的X、Y、Z方向上的影响也会发生改变。磁场传感器会返回三个数据，分别代表周围磁场分解到X、Y、Z三个方向的磁场分量，磁场数据的单位是微特斯拉。

重力传感器

重力传感器会返回一个三维向量，这个三维向量可显示重力的方向和强度。重力传感器的坐标系统和加速度传感器的坐标系统相同。

线性加速度传感器

线性加速度传感器返回一个三维向量显示设备在各个方向的加速度（不包含重力加速度）。线性加速度传感器的坐标系统和加速度传感器的坐标系统相同。

线性加速度传感器、重力传感器、加速度传感器，这三者输出值的关系如下：

加速度传感器 = 重力传感器　+　线性加速度传感器。

温度传感器

温度传感器用于获取设备所处环境的温度。温度传感器会返回一个数据，代表手机设备周围的温度，单位是摄氏度。

光传感器

光传感器用于获取设备周围光的强度。光传感器会返回一个数据，代表手机周围光的强度，单位是勒克斯。

压力传感器

压力传感器用于获取设备周围压力的大小。压力传感器会返回一个数据，代表设备周围压力的大小。

心率传感器

心率传感器是在5.0之后新增的一个传感器，用于返回佩戴设备的人每分钟的心跳次数。该传感器返回的数据准确性可以通过SensorEvent的accuracy进行判断，如果该属性值为：SENSOR\_STATUS\_UNRELIABLE或SENSOR\_STATUS\_NO\_CONTACT，则表明传感器返回的数据是不太可靠的，应该丢弃。

在使用心率传感器时，需要增加如下权限：

<uses-permission android:name="android.permission.BODY\_SENSORS"/>

实例（获取各传感器数据并展示）

public class MainActivity extends AppCompatActivity implements SensorEventListener{

private SensorManager mSensorManager;

private TextView mTxtValue1;

private TextView mTxtValue2;

private TextView mTxtValue3;

private TextView mTxtValue4;

private TextView mTxtValue5;

private TextView mTxtValue6;

private TextView mTxtValue7;

private TextView mTxtValue8;

private TextView mTxtValue9;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mTxtValue1 = (TextView) findViewById(R.id.txt\_value1);

mTxtValue2 = (TextView) findViewById(R.id.txt\_value2);

mTxtValue3 = (TextView) findViewById(R.id.txt\_value3);

mTxtValue4 = (TextView) findViewById(R.id.txt\_value4);

mTxtValue5 = (TextView) findViewById(R.id.txt\_value5);

mTxtValue6 = (TextView) findViewById(R.id.txt\_value6);

mTxtValue7 = (TextView) findViewById(R.id.txt\_value7);

mTxtValue8 = (TextView) findViewById(R.id.txt\_value8);

mTxtValue9 = (TextView) findViewById(R.id.txt\_value9);

// 获取传感器管理对象

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

}

@Override

protected void onResume() {

super.onResume();

// 为加速度传感器注册监听器

mSensorManager.registerListener(this, mSensorManager.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER), SensorManager.SENSOR\_DELAY\_GAME);

// 为方向传感器注册监听器

mSensorManager.registerListener(this, mSensorManager.getDefaultSensor(Sensor.TYPE\_ORIENTATION), SensorManager.SENSOR\_DELAY\_GAME);

// 为陀螺仪传感器注册监听器

mSensorManager.registerListener(this, mSensorManager.getDefaultSensor(Sensor.TYPE\_GYROSCOPE), SensorManager.SENSOR\_DELAY\_GAME);

// 为磁场传感器注册监听器

mSensorManager.registerListener(this, mSensorManager.getDefaultSensor(Sensor.TYPE\_MAGNETIC\_FIELD), SensorManager.SENSOR\_DELAY\_GAME);

// 为重力传感器注册监听器

mSensorManager.registerListener(this, mSensorManager.getDefaultSensor(Sensor.TYPE\_GRAVITY), SensorManager.SENSOR\_DELAY\_GAME);

// 为线性加速度传感器注册监听器

mSensorManager.registerListener(this, mSensorManager.getDefaultSensor(Sensor.TYPE\_LINEAR\_ACCELERATION), SensorManager.SENSOR\_DELAY\_GAME);

// 为温度传感器注册监听器

mSensorManager.registerListener(this, mSensorManager.getDefaultSensor(Sensor.TYPE\_AMBIENT\_TEMPERATURE), SensorManager.SENSOR\_DELAY\_GAME);

// 为光传感器注册监听器

mSensorManager.registerListener(this, mSensorManager.getDefaultSensor(Sensor.TYPE\_LIGHT), SensorManager.SENSOR\_DELAY\_GAME);

// 为压力传感器注册监听器

mSensorManager.registerListener(this, mSensorManager.getDefaultSensor(Sensor.TYPE\_PRESSURE), SensorManager.SENSOR\_DELAY\_GAME);

}

@Override

protected void onStop() {

super.onStop();

// 取消监听

mSensorManager.unregisterListener(this);

}

// 当传感器的值改变的时候回调该方法

@Override

public void onSensorChanged(SensorEvent event) {

float[] values = event.values;

// 获取传感器类型

int type = event.sensor.getType();

StringBuilder sb;

switch (type){

case Sensor.TYPE\_ACCELEROMETER:

sb = new StringBuilder();

sb.append("加速度传感器返回数据：");

sb.append("\nX方向的加速度：");

sb.append(values[0]);

sb.append("\nY方向的加速度：");

sb.append(values[1]);

sb.append("\nZ方向的加速度：");

sb.append(values[2]);

mTxtValue1.setText(sb.toString());

break;

case Sensor.TYPE\_ORIENTATION:

sb = new StringBuilder();

sb.append("\n方向传感器返回数据：");

sb.append("\n绕Z轴转过的角度：");

sb.append(values[0]);

sb.append("\n绕X轴转过的角度：");

sb.append(values[1]);

sb.append("\n绕Y轴转过的角度：");

sb.append(values[2]);

mTxtValue2.setText(sb.toString());

break;

case Sensor.TYPE\_GYROSCOPE:

sb = new StringBuilder();

sb.append("\n陀螺仪传感器返回数据：");

sb.append("\n绕X轴旋转的角速度：");

sb.append(values[0]);

sb.append("\n绕Y轴旋转的角速度：");

sb.append(values[1]);

sb.append("\n绕Z轴旋转的角速度：");

sb.append(values[2]);

mTxtValue3.setText(sb.toString());

break;

case Sensor.TYPE\_MAGNETIC\_FIELD:

sb = new StringBuilder();

sb.append("\n磁场传感器返回数据：");

sb.append("\nX轴方向上的磁场强度：");

sb.append(values[0]);

sb.append("\nY轴方向上的磁场强度：");

sb.append(values[1]);

sb.append("\nZ轴方向上的磁场强度：");

sb.append(values[2]);

mTxtValue4.setText(sb.toString());

break;

case Sensor.TYPE\_GRAVITY:

sb = new StringBuilder();

sb.append("\n重力传感器返回数据：");

sb.append("\nX轴方向上的重力：");

sb.append(values[0]);

sb.append("\nY轴方向上的重力：");

sb.append(values[1]);

sb.append("\nZ轴方向上的重力：");

sb.append(values[2]);

mTxtValue5.setText(sb.toString());

break;

case Sensor.TYPE\_LINEAR\_ACCELERATION:

sb = new StringBuilder();

sb.append("\n线性加速度传感器返回数据：");

sb.append("\nX轴方向上的线性加速度：");

sb.append(values[0]);

sb.append("\nY轴方向上的线性加速度：");

sb.append(values[1]);

sb.append("\nZ轴方向上的线性加速度：");

sb.append(values[2]);

mTxtValue6.setText(sb.toString());

break;

case Sensor.TYPE\_AMBIENT\_TEMPERATURE:

sb = new StringBuilder();

sb.append("\n温度传感器返回数据：");

sb.append("\n当前温度为：");

sb.append(values[0]);

mTxtValue7.setText(sb.toString());

break;

case Sensor.TYPE\_LIGHT:

sb = new StringBuilder();

sb.append("\n光传感器返回数据：");

sb.append("\n当前光的强度为：");

sb.append(values[0]);

mTxtValue8.setText(sb.toString());

break;

case Sensor.TYPE\_PRESSURE:

sb = new StringBuilder();

sb.append("\n压力传感器返回数据：");

sb.append("\n当前压力为：");

sb.append(values[0]);

mTxtValue9.setText(sb.toString());

break;

}

}

// 当传感器精度发生改变时回调该方法

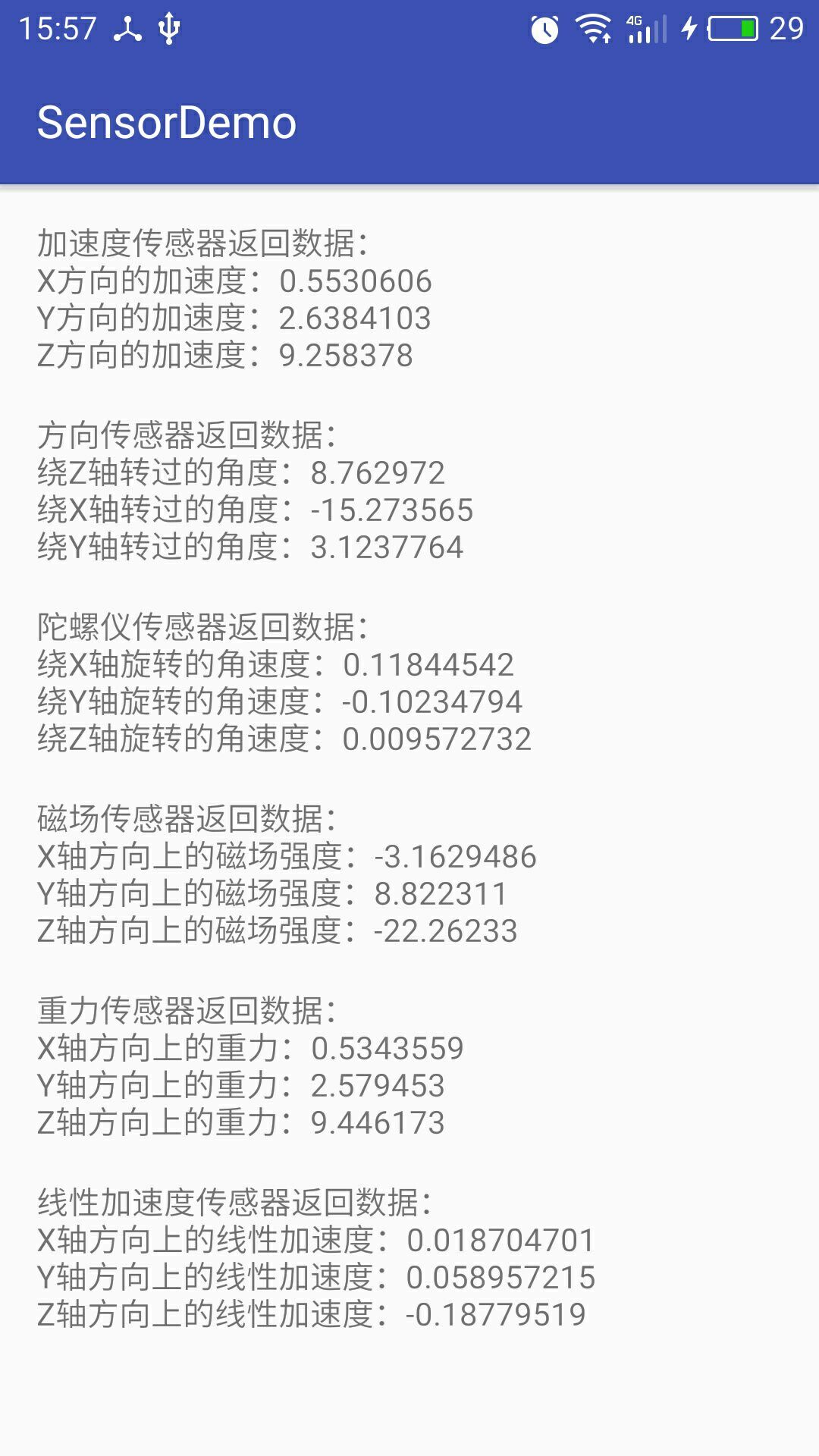
@Override

public void onAccuracyChanged(Sensor sensor, int accuracy) {

}

}

运行结果：



https://blog.csdn.net/airsaid/article/details/52902299