Собираем показания датчиков

* [Общая информация](http://developer.alexanderklimov.ru/android/sensors.php#common)
* [Таблица значений, возвращаемых датчиками](http://developer.alexanderklimov.ru/android/sensors.php#table)
* [Огласите весь список, пожалуйста!](http://developer.alexanderklimov.ru/android/sensors.php#getall)
* [Интерфейс SensorEventListener - отслеживаем показания](http://developer.alexanderklimov.ru/android/sensors.php#sensoreventlistener)

Общая информация

Датчики, следящие за физическими свойствами и состоянием окружающей среды, предоставляют инновационные способы для улучшения мобильных приложений. Наличие в современных телефонах электронных компасов, датчиков равновесия, яркости и близости открывает целый ряд новых возможностей для взаимодействия с устройством, таких как дополненная реальность и ввод данных, основанный на перемещениях в пространстве.

Датчики в Android делятся на несколько категорий: движения, положения и окружающей среды. Ниже перечислены некоторые виды популярных датчиков:

* Акселерометр (TYPE\_ACCELEROMETER)
* Гироскоп (TYPE\_GYROSCOPE)
* Датчик освещения (TYPE\_LIGHT)
* Датчик расстояния (TYPE\_PROXIMITY)
* Датчик магнитных полей (TYPE\_MAGNETIC\_FIELD)
* Барометр (TYPE\_PRESSURE)
* Датчик температуры окружающей среды (TYPE\_AMBIENT\_TEMPERATURE)
* Измеритель относительной влажности (TYPE\_RELATIVE\_HUMIDITY)

В каждом телефоне может быть свой набор датчиков. В большинстве аппаратов есть — акселерометр и гироскоп.

Каждый из представленных датчиков заслуживает отдельной статьи. Имейте в виду, что существуют устаревшие классы для работы с датчиками, в частности, для датчиков ориентации и температуры.

Необходимо помнить несколько вещей, работая с датчиками:

* Показания бывают очень неровными. Вам нужно использовать какое-то среднее значение показаний, но не переборщить, чтобы приложение оставалось отзывчивым
* Данные приходят неравномерно. Не ждите спокойного, ровного потока данных
* Попробуйте предугадать будущие действия пользователя. Например, если идут данные о начале вращения устройства, можно предугадать следующее движение и подготовиться к нему

На эмуляторе практически невозможно тестировать работу с датчиками, поэтому используйте реальные устройства. В последних версиях эмуляторов список возможностей датчиков расширился. Смотрите в настройках эмулятора раздел **Virtual sensors**.

За работу с сенсорами отвечает класс **SensorManager**, содержащий несколько констант, которые характеризуют различные аспекты системы датчиков Android, в том числе:

**Тип датчика**

Ориентация, акселерометр, свет, магнитное поле, близость, температура и т.д.

**Частота измерений**

Максимальная, для игр, обычная, для пользовательского интерфейса. Когда приложение запрашивает конкретное значение частоты отсчетов, с точки зрения сенсорной подсистемы это лишь рекомендация. Никакой гарантии, что измерения будут производиться с указанной частотой, нет.

**Точность**

Высокая, низкая, средняя, ненадежные данные.

Типы датчиков

* [TYPE\_ACCELEROMETER](http://developer.alexanderklimov.ru/android/sensor-accelerometer.php) - Измеряет ускорение в пространстве по осям X, Y, Z
* [TYPE\_AMBIENT\_TEMPERATURE](http://developer.alexanderklimov.ru/android/sensor-temperature.php) - Новый датчик для измерения температуры (API 14) в градусах Цельсия, который заменил устаревший **TYPE\_TEMPERATURE**
* **TYPE\_GRAVITY** - Трёхосевой датчик силы тяжести. Как правило, это виртуальный датчик и представляет собой низкочастотный фильтр для показаний, возвращаемых акселерометром
* [TYPE\_GYROSCOPE](http://developer.alexanderklimov.ru/android/sensor-gyroscope.php) - Трёхосевой гироскоп, возвращающий текущее положение устройства в пространстве в градусах по трём осям. По другим данным, возвращает скорость вращения устройства по трём осям в радианах в секунду.
* [TYPE\_LIGHT](http://developer.alexanderklimov.ru/android/sensor-light.php) - Измеряет степень освещенности. Датчик окружающей освещенности, который описывает внешнюю освещенность в люксах. Этот тип датчиков обычно используется для динамического изменения яркости экрана.
* **TYPE\_LINEAR\_ACCELERATION** - Трёхосевой датчик линейного ускорения, возвращающий показатели ускорения без учёта силы тяжести. Это виртуальный датчик, использующий показания акселерометра.
* [TYPE\_MAGNETIC\_FIELD](http://developer.alexanderklimov.ru/android/sensor-magnetic-field.php) - Датчик магнитного поля, определяющий текущие показатели магнитного поля в микротеслах по трём осям.
* **TYPE\_ORIENTATION** - Датчик ориентации. Измеряет повороты, наклоны и вращение устройства
* [TYPE\_PRESSURE](http://developer.alexanderklimov.ru/android/sensor-barometer.php) - Датчик атмосферного давления (барометр), возвращающий текущее давление в миллибарах. Можно определять высоту над уровнем моря, путём сравнения атмосферного давления в двух точках. Также барометры могут применяться для прогнозирования погоды.
* [TYPE\_PROXIMITY](http://developer.alexanderklimov.ru/android/sensor-proximity.php) - Датчик приближенности, который сигнализирует о расстоянии между устройством и целевым объектом в сантиметрах. Каким образом выбирается объект и какие расстояния поддерживаются, зависит от аппаратной реализации данного датчика, возможно возвращение двух значений - Близко и Далеко. Типичное его применение — обнаружение расстояния между устройством и ухом пользователя для автоматического регулирования яркости экрана или выполнения голосовой команды.
* [TYPE\_RELATIVE\_HUMIDITY](http://developer.alexanderklimov.ru/android/sensor-humidity.php) - Датчик относительной влажности в виде процентного значения (API 14)
* [TYPE\_ROTATION\_VECTOR](http://developer.alexanderklimov.ru/android/sensor-rotation-vector.php) - Возвращает положение устройства в пространстве в виде угла относительно оси. Виртуальный датчик, берущий показания от акселерометра и гироскопа. Также может использовать показания датчика магнитного поля
* [TYPE\_STEP\_COUNTER](http://developer.alexanderklimov.ru/android/sensor-stepcounter.php) - датчик для подсчёта количества шагов

Кроме аппаратных датчиков, в устройствах используются виртуальные датчики, которые предоставляют упрощенные, уточнённые или комбинированные показания, используя комбинацию из нескольких аппаратных датчиков. В некоторых случаях этот способ удобнее.

Чтобы получить доступ к сенсорам, нужно вызвать метод **getSystemService()**.

private SensorManager sensorManager;

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

Устройство может включать в себя несколько реализаций одного и того же типа датчиков. Чтобы найти реализацию, используемую по умолчанию, вызовите метод **getDefaultSensor()** из объекта **SensorManager**, передавая ему в качестве параметра тип датчика в виде одной из констант, описанных выше.

Следующий фрагмент кода вернёт объект, описывающий гироскоп по умолчанию. Если для данного типа не существует датчика по умолчанию, будет возвращено значение *null*.

Sensor defaultGyroscope = sensorManager.getDefaultSensor(Sensor.TYPE\_GYROSCOPE);

Таблица значений, возвращаемых датчиками

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип датчика** | **Количество значений** | **Содержание значений** | **Примечание** |
| TYPE\_ACCELEROMETER | 3 | value[0]:ось X (поперечная) value[1]: ось Y (продольная) value[2]:ось Y (вертикальная) | Ускорение (м/с2) по трём осям. Константы SensorManager.GRAVITY\_\* |
| TYPE\_GRAVITY | 3 | value[0]:ось X (поперечная) value[1]: ось Y (продольная) value[2]:ось Y (вертикальная) | Сила тяжести (м/с2) по трём осям. Константы SensorManager.GRAVITY\_\* |
| TYPE\_RELATIVE\_HUMIDITY | 1 | value[0]:относительная влажность | Относительная влажность в процентах (%) |
| TYPE\_LINEAR\_ACCELERATION | 3 | value[0]:ось X (поперечная) value[1]: ось Y (продольная) value[2]:ось Y (вертикальная) | Линейное ускорение (м/с2) по трём осям без учёта силы тяжести |
| TYPE\_GYROSCOPE | 3 | value[0]:ось X value[1]:ось Y value[2]:ось Z | Скорость вращения (рад/с) по трём осям |
| TYPE\_ROTATION\_VECTOR | 3 (+1 необяз.) | values[0]:x\*sin(q/2) values[1]:y\*sin(q/2) values[2]:z\*sin(q/2) values[3]:cos(q/2) (необяз.) | Положение устройства в пространстве. Описывается в виде угла поворота относительно оси в градусах |
| TYPE\_MAGNETIC\_FIELD | 3 | value[0]:ось X (поперечная) value[1]: ось Y (продольная) value[2]:ось Y (вертикальная) | Внешнее магнитное поле (мкТл) |
| TYPE\_LIGHT | 1 | value[0]:освещённость | Внешняя освещённость (лк). Константы SensorManager.LIGHT\_\* |
| TYPE\_PRESSURE | 1 | value[0]:атм.давление | Атмосферное давление (мбар) |
| TYPE\_PROXIMITY | 1 | value[0]:расстояние | Расстояние до цели |
| TYPE\_AMBIENT\_TEMPERATURE | 1 | value[0]:температура | Температура воздухах в градусах по Цельсию |

Огласите весь список, пожалуйста!

У класса **SensorManager** есть метод **getSensorList()**, позволяющий получить список доступных датчиков на устройстве через константу **Sensor.TYPE\_ALL** и метод **getName()**:

package ru.alexanderklimov.sensors;

import java.util.ArrayList;

import java.util.List;

import android.app.ListActivity;

import android.content.Context;

import android.hardware.Sensor;

import android.hardware.SensorManager;

import android.os.Bundle;

import android.widget.ArrayAdapter;

public class SensorsActivity extends ListActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

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

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

List<String> listSensorType = new ArrayList<>();

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

listSensorType.add(listSensor.get(i).getName());

}

setListAdapter(new ArrayAdapter<>(this,

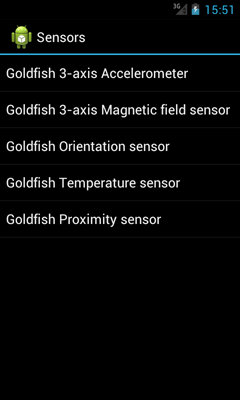
android.R.layout.simple\_list\_item\_1, listSensorType));

getListView().setTextFilterEnabled(true);

}

}

Так как у каждого устройства свой набор датчиков, то результаты будут у всех разными. Ниже приведены скриншоты с эмулятора и с реального устройства. В первом случае вывелось только 5 датчиков, во-втором было гораздо больше - вы видите только ту часть, которая поместилась на экране.

Также можно использовать метод **getSensorList()** для получения списка доступных датчиков конкретного типа. В следующем фрагменте кода будут возвращены объекты **Sensor**, представляющие собой все доступные датчики давления:

List<Sensor> pressureSensors = sensorManager.getSensorList(Sensor.TYPE\_PRESSURE);

Интерфейс SensorEventListener - отслеживаем показания

Также вам понадобится интерфейс **android.hardware.SensorListener**. Интерфейс реализован с помощью класса, который используется для ввода значений датчиков по мере их изменения в режиме реального времени. Приложение реализует этот интерфейс для мониторинга одного или нескольких имеющихся аппаратных датчиков.

Интерфейс включает в себя два необходимых метода:

* Метод **onSensorChanged(int sensor, float values[])** вызывается всякий раз, когда изменяется значение датчика. Этот метод вызывается только для датчиков, контролируемых данным приложением. В число аргументов метода входит целое, которое указывает, что значение датчика изменилось, и массив значений с плавающей запятой, отражающих собственно значение датчика. Некоторые датчики выдают только одно значение данных, тогда как другие предоставляют три значения с плавающей запятой. Датчики ориентации и акселерометр дают по три значения данных каждый.
* Метод **onAccuracyChanged(int sensor,int accuracy)** вызывается при изменении точности показаний датчика. Аргументами служат два целых числа: одно указывает датчик, а другое соответствует новому значению точности этого датчика.

Служба датчиков вызывает **onSensorChanged()** каждый раз при изменении значений. Все датчики возвращают массив значений с плавающей точкой. Размер массива зависит от особенностей датчика. Датчик **TYPE\_TEMPERATURE** возвращает одно значение - температуру в градусах Цельсия, другие могут возвращать несколько значений. Вы можете использовать только нужные значения. Например, для получения сведений только о магнитном азимуте достаточно использовать первое числов, возвращаемое датчиком **TYPE\_ORIENTATION**.

Параметр **accuracy**, используемый в методах для представления степени точности датчика, использует одну из констант

* **SensorManager.SENSOR\_STATUS\_ACCURACY\_LOW**. Говорит о том, что данные, предоставляемые датчиком, имеют низкую точность и нуждаются в калибровке.
* **SensorManager.SENSOR\_STATUS\_ACCURACY\_MEDIUM**. Говорит о средней степени точности датчика и том, что калибровка может улучшить результат.
* **SensorManager.SENSOR\_STATUS\_ACCURACY\_HIGH**. Показатели датчика точны настолько, насколько это возможно.
* **SensorManager.SENSOR\_STATUS\_UNRELIABLE**. Данные, предоставляемые датчиком, недостоверны. Это значит, что датчик необходимо откалибровать, иначе невозможно считывать результаты.

Чтобы получать события, генерируемые датчиками, зарегистрируйте свою реализацию интерфейса **SensorEventListener** с помощью **SensorManager**. Укажите объект **Sensor**, за которым вы хотите наблюдать, и частоту, с которой вам необходимо получать обновления.

После получения объекта вы вызываете метод **registerListener()** в методе **onResume()**, чтобы начать получать обновлённые данные, и вызываете **unregisteredListener()** в методе **onPause()**, чтобы остановить получение данных. В этом случае датчики будут использоваться только тогда, когда активность видна на экране.

В следующем примере показан процесс регистрации SensorEventListener для датчика приближенности по умолчанию с указанием стандартной частоты обновления:

Sensor sensor = sensorManager.getDefaultSensor(Sensor.TYPE\_PROXIMITY);

sensorManager.registerListener(mySensorEventListener,

sensor,

SensorManager.SENSOR\_DELAY\_NORMAL);

Класс SensorManager содержит следующие константы для выбора подходящей частоты обновлений (в порядке убывания):

* **SensorManager.SENSOR\_DELAY\_FASTEST** — самая высокая возможная частота обновления показаний датчиков;
* **SensorManager.SENSOR\_DELAY\_GAME** — частота, используемая для управления играми;
* **SensorManager.SENSOR\_DELAY\_NORMAL** — частота обновлений по умолчанию;
* **SensorManager.SENSOR\_DELAY\_UI** — частота для обновления пользовательского интерфейса.

Выбранная вами частота необязательно будет соблюдаться. **SensorManager** может возвращать результаты быстрее или медленней, чем вы указали (хотя, как правило, это происходит быстрее). Чтобы минимизировать расход ресурсов при использовании датчиков в приложении, необходимо пытаться подбирать наиболее низкую частоту.

Следите за обновлениями в документации

API для датчиков постоянно пополняется. Если посмотреть на страницу документации по классу **Sensor**, то можно увидеть, что новые методы появлялись в API версии 14, 18, 19, 20, 21, 24, 26.