Министерство образования и науки России

**ФГБО УВО "Севастопольский государственный университет"**

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

к лабораторной работе №4

**"Исследование датчиков в ОС Android"**

по дисциплине

"**МИТ**"

для студентов основного профиля

09.03.02 "Информационные системы и технологии"

всех форм обучения.

**Севастополь**

**2018**

**1. Цель работы**

Исследование процесса разработки практического мобильного приложения с использованием датчиков.

# 2. Краткие теоретические сведения

Android, как система, ориентированная на мобильный сегмент рынка, имеет мощную поддержку разнообразных датчиков, а также датчика GPS. Далее мы рассмотрим два датчика – акселерометр (датчик ускорения) и GPS датчик географических координат.

Для выполнения работы необходимы базовые знания языка Java и Android API. Самое важное требование - необходимо Android- устройство (смартфон или планшет) с акселерометром и GPS.

Современные мобильные телефоны — не просто устройства для связи, подключенные к Интернету. Смартфоны с микрофонами, камерами, акселерометрами, компасами, датчиками температуры и яркости способны «воспринимать» окружающий мир, расширяя ваше собственное восприятие.

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.

В этой работе вы познакомитесь с датчиками в Android и объектом SеnsorManagcr, отслеживающим их состояние. Подробно рассмотрите возможности акселерометра и датчиков ориентации, научитесь использовать их для определения изменений положения и ускорения устройства. Это может быть особенно полезно при создании интерфейсов, основанных на движении, это позволит добавлять новые измерения к вашим геолокационным приложениям.

Чтобы начать работать с датчиком, нужно определить его тип. Удобнее всего это сделать с помощью класса Sensor, так как в нем уже определены все типы сенсоров в виде констант. Рассмотрим их подробнее:

Sensor.TYPE\_ACCELEROMETER — трехосевой акселерометр, возвращающий ускорение по трем осям (в метрах в секунду в квадрате). Связанная система координат представлена на рис. 1.

Sensor.TYPE\_LIGHT — датчик освещенности, возвращающий значение в люксах, обычно используется для динамического изменения яркости экрана. Также для удобства степень освещенности можно получить в виде характеристик — «темно», «облачно», «солнечно» (к этому мы еще вернемся).

Sensor.TYPE\_AMBIENT\_TEMPERATURE — термометр, возвращает температуру окружающей среды в градусах Цельсия.

Sensor.TYPE\_PROXIMITY — датчик приближенности, который сигнализирует о расстоянии между устройством и пользователем (в сантиметрах). Когда в момент разговора гаснет экран — срабатывает именно этот датчик. На некоторых девайсах возвращается только два значения: «далеко» и «близко».

Sensor.TYPE\_GYROSCOPE — трехосевой гироскоп, возвращающий скорость вращения устройства по трем осям (радиан в секунду).

Sensor.TYPE\_MAGNETIC\_FIELD — магнитометр, определяющий показания магнитного поля в микротеслах (мкТл) по трем осям (имеется в смартфонах с аппаратным компасом).

Sensor.TYPE\_PRESSURE — датчик атмосферного давления (по-простому — барометр), который возвращает текущее атмосферное давление в миллибарах (мбар). Если немного вспомнить физику, то, используя значение этого датчика, можно легко вычислить высоту (а ежели вспоминать ну никак не хочется, можно воспользоваться готовым методом getAltitude из объекта SensorManager).

Sensor.TYPE\_RELATIVE\_HUMIDITY — датчик относительной влажности в процентах. Sensor.TYPE\_STEP\_COUNTER (с API 19) — счетчик шагов с момента включения устройства (обнуляется только после перезагрузки).

Sensor.TYPE\_MOTION\_DETECT (с API 24) — детектор движения смартфона. Если устройство находится в движении от пяти до десяти секунд, возвращает единицу (по всей видимости, задел для аппаратной функции «антивор»).

Sensor.TYPE\_HEART\_BEAT (с API 24) — детектор биения сердца.

Sensor.TYPE\_HEART\_RATE (с API 20) — датчик, возвращающий пульс (ударов в минуту). Этот датчик примечателен тем, что требует явного разрешения android.permission.BODY\_SENSORS в манифесте.

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

Чтобы узнать, какие сенсоры есть в смартфоне, следует использовать метод **getSensorList**объекта **SensorManager**:

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

Полученный список будет включать все поддерживаемые датчики: как аппаратные, так и виртуальные.

Для получения списка всех доступных датчиков конкретного типа необходимо указать соответствующую константу. Например, следующее:

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

Вернет все доступные барометрические датчики. Причем аппаратные реализации окажутся в начале списка, а виртуальные — в конце.

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

Рассмотрим пример программы, которая будет выводить список доступных сенсоров и данные с сенсора света.

В **strings.xml** добавим строки:

|  |  |
| --- | --- |
|  | <string name="list">List</string>  <string name="light">Light</string> |

Экран **main.xml**:

|  |  |
| --- | --- |
|  | <?xml version="1.0" encoding="utf-8"?>  <RelativeLayout  xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">  <Button  android:id="@+id/btnSensList"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:onClick="onClickSensList"  android:text="@string/list">  </Button>  <Button  android:id="@+id/btnSensLight"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_toRightOf="@id/btnSensList"  android:onClick="onClickSensLight"  android:text="@string/light">  </Button>  <ScrollView  android:id="@+id/scroll"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_below="@id/btnSensList">  <TextView  android:id="@+id/tvText"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content">  </TextView>  </ScrollView>  </RelativeLayout> |

Кнопки необходимы для получения списка сенсоров и данных по свету; TextView для отображения данных.

**MainActivity.java**:

|  |  |
| --- | --- |
|  | package ru.startandroid.develop.p1371sensors;  import java.util.List;  import android.app.Activity;  import android.hardware.Sensor;  import android.hardware.SensorEvent;  import android.hardware.SensorEventListener;  import android.hardware.SensorManager;  import android.os.Bundle;  import android.view.View;  import android.widget.TextView;  public class MainActivity extends Activity {  TextView tvText;  SensorManager sensorManager;  List<Sensor> sensors;  Sensor sensorLight;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.main);  tvText = (TextView) findViewById(R.id.tvText);  sensorManager = (SensorManager) getSystemService(SENSOR\_SERVICE);  sensors = sensorManager.getSensorList(Sensor.TYPE\_ALL);  sensorLight = sensorManager.getDefaultSensor(Sensor.TYPE\_LIGHT);  }  public void onClickSensList(View v) {  sensorManager.unregisterListener(listenerLight, sensorLight);  StringBuilder sb = new StringBuilder();  for (Sensor sensor : sensors) {  sb.append("name = ").append(sensor.getName())  .append(", type = ").append(sensor.getType())  .append("\nvendor = ").append(sensor.getVendor())  .append(" ,version = ").append(sensor.getVersion())  .append("\nmax = ").append(sensor.getMaximumRange())  .append(", resolution = ").append(sensor.getResolution())  .append("\n--------------------------------------\n");  }  tvText.setText(sb);  }  public void onClickSensLight(View v) {  sensorManager.registerListener(listenerLight, sensorLight,  SensorManager.SENSOR\_DELAY\_NORMAL);  }  @Override  protected void onPause() {  super.onPause();  sensorManager.unregisterListener(listenerLight, sensorLight);  }  SensorEventListener listenerLight = new SensorEventListener() {  @Override  public void onAccuracyChanged(Sensor sensor, int accuracy) {  }  @Override  public void onSensorChanged(SensorEvent event) {  tvText.setText(String.valueOf(event.values[0]));  }  };  } |

В **onCreate** получаем [SensorManager](http://developer.android.com/reference/android/hardware/SensorManager.html). У него запрашиваем полный список сенсоров, используя метод [getSensorList](http://developer.android.com/reference/android/hardware/SensorManager.html#getSensorList(int)) и передавая туда тип сенсора [TYPE\_ALL](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_ALL).

Чтобы получить конкретный сенсор ([Sensor](http://developer.android.com/reference/android/hardware/Sensor.html)), вызываем метод [getDefaultSensor](http://developer.android.com/reference/android/hardware/SensorManager.html#getDefaultSensor(int)). Передаем тип [TYPE\_LIGHT](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_LIGHT) и получаем сенсор света. Необходимо быть аккуратнее, так как если такого сенсора в девайсе нет – то метод вернет null.

В **onClickSensList** отписываем слушателя от сенсора.

Далее берем список сенсоров и выводим по ним информацию на экран:  
[getName](http://developer.android.com/reference/android/hardware/Sensor.html#getName()) – имя;  
[getType](http://developer.android.com/reference/android/hardware/Sensor.html#getType()) – тип;  
[getVendor](http://developer.android.com/reference/android/hardware/Sensor.html#getVendor()) – создатель;  
[getVersion](http://developer.android.com/reference/android/hardware/Sensor.html#getVersion()) – версия;  
[getMaximumRange](http://developer.android.com/reference/android/hardware/Sensor.html#getMaximumRange()) – максимальное значение, которое может вернуть сенсор;  
[getResolution](http://developer.android.com/reference/android/hardware/Sensor.html#getResolution()) – это минимальный шаг, с которым может изменяться значение.

В методе **onClickSensLight** используем метод [registerListener](http://developer.android.com/reference/android/hardware/SensorManager.html#registerListener(android.hardware.SensorEventListener,%20android.hardware.Sensor,%20int)), чтобы на ранее полученный сенсор света (sensorLight) повесить своего слушателя listenerLight. Третий параметр метода – скорость получения новых данных. Т.е. насколько часто вам необходимо получать данные от сенсора. Есть 4 скорости в порядке убывания: [SENSOR\_DELAY\_NORMAL](http://developer.android.com/reference/android/hardware/SensorManager.html#SENSOR_DELAY_NORMAL),  [SENSOR\_DELAY\_UI](http://developer.android.com/reference/android/hardware/SensorManager.html#SENSOR_DELAY_UI),  [SENSOR\_DELAY\_GAME](http://developer.android.com/reference/android/hardware/SensorManager.html#SENSOR_DELAY_GAME),  [SENSOR\_DELAY\_FASTEST](http://developer.android.com/reference/android/hardware/SensorManager.html#SENSOR_DELAY_FASTEST).

Help сообщает, что система может проигнорить это значение и выдавать данные как ей удобно. А начиная с API Level 9 можно вместо константы скорости передавать свое значение в микросекундах. Не перепутайте с миллисекундами.

В **onPause** отписываем своего слушателя от сенсора света. Тут, как обычно, рекомендуется отписываться, как только данные вам не нужны, чтобы не расходовать зря батарею.

**listenerLight** – слушатель, реализует интерфейс [SensorEventListener](http://developer.android.com/reference/android/hardware/SensorEventListener.html). У него два метода:

[onAccuracyChanged](http://developer.android.com/reference/android/hardware/SensorEventListener.html#onAccuracyChanged(android.hardware.Sensor,%20int)) – вызывается, когда меняется точность данных сенсора и в начале получения данных. Дает нам объект-сенсор и уровень точности:

[SENSOR\_STATUS\_ACCURACY\_HIGH](http://developer.android.com/reference/android/hardware/SensorManager.html#SENSOR_STATUS_ACCURACY_HIGH) – максимально возможная точность;  
[SENSOR\_STATUS\_ACCURACY\_MEDIUM](http://developer.android.com/reference/android/hardware/SensorManager.html#SENSOR_STATUS_ACCURACY_MEDIUM) – средняя точность, калибровка могла бы улучшить результат;  
[SENSOR\_STATUS\_ACCURACY\_LOW](http://developer.android.com/reference/android/hardware/SensorManager.html#SENSOR_STATUS_ACCURACY_LOW) – низкая точность, необходима калибровка;  
[SENSOR\_STATUS\_UNRELIABLE](http://developer.android.com/reference/android/hardware/SensorManager.html#SENSOR_STATUS_UNRELIABLE) – данные сенсора совсем ни о чем. Либо нужна калибровка, либо невозможно чтение данных.

[onSensorChanged](http://developer.android.com/reference/android/hardware/SensorEventListener.html#onSensorChanged(android.hardware.SensorEvent)) – здесь то мы и получаем данные от сенсора в объекте [SensorEvent](http://developer.android.com/reference/android/hardware/SensorEvent.html).

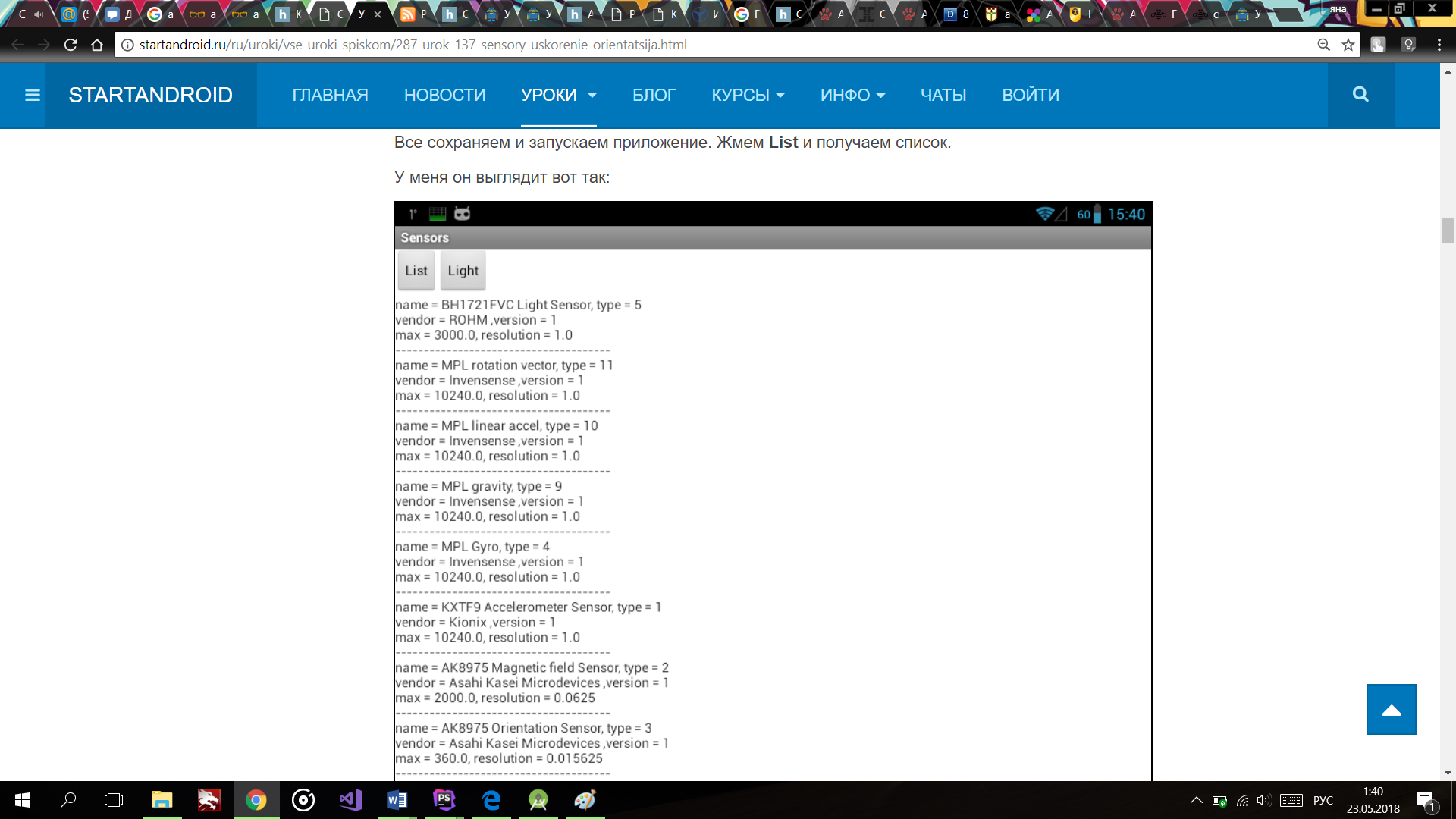


Рисунок 2.1 – Результат полученных данных

**Датчик акселерометра**. Для работы с обычными датчиками в Android API предназначен пакет android.hardware, содержащий такие классы как Sensor, SensorManager и **SensorEventListener.**

Акселерометр – это прибор, предназначенный для измерения кажущегося ускорения. Кажущееся ускорение – это разница между гравитационным и истинным ускорениями объекта.

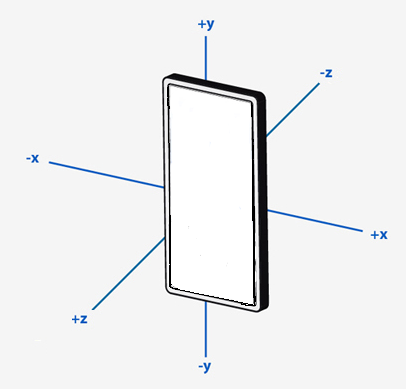


Рисунок 2.2 – Наглядное представление гироскопа.

**Гироскоп**. Гироскоп измеряет угловую скорость или частоту вращения вдоль трех осей. Все значения исчисляются в рад/с. Вращения против часовой стрелки имеют положительные значения. Поскольку речь идет об угловой скорости, для расчета угла необходимо интегрировать значение в течении некоторого периода времени.

Датчик ориентации — это комбинация датчика магнитного поля, выполняющего роль электронного компаса, и акселерометра, измеряющего наклон и вращение.

Далее представлен пример приложения, которое следит за изменениями значений датчиков ориентации и акселерометра

package com.example.mymsi.mit;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.app.Activity;  
import android.os.Bundle;  
import android.util.Log;  
import android.widget.TextView;  
import android.hardware.SensorManager;  
import android.hardware.SensorListener;  
public class IBMEyes extends Activity implements SensorListener {  
 final String tag = "IBMEyes";  
 SensorManager sm = null;  
 TextView xViewA = null;  
 TextView yViewA = null;  
 TextView zViewA = null;  
 TextView xViewO = null;  
 TextView yViewO = null;  
 TextView zViewO = null;  
  
 */\*\* Called when the activity is first created. \*/* @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 *// get reference to SensorManager* sm = (SensorManager) getSystemService(*SENSOR\_SERVICE*);  
 xViewA = (TextView) findViewById(R.id.*xbox*);  
 yViewA = (TextView) findViewById(R.id.*ybox*);  
 zViewA = (TextView) findViewById(R.id.*zbox*);  
 xViewO = (TextView) findViewById(R.id.*xboxo*);  
 yViewO = (TextView) findViewById(R.id.*yboxo*);  
 zViewO = (TextView) findViewById(R.id.*zboxo*);  
 }  
 public void onSensorChanged(int sensor, float[] values) {  
 synchronized (this) {  
 Log.*d*(tag, "onSensorChanged: " + sensor + ", x: " +  
 values[0] + ", y: " + values[1] + ", z: " + values[2]);  
 if (sensor == SensorManager.*SENSOR\_ORIENTATION*) {  
 xViewO.setText("Orientation X: " + values[0]);  
 yViewO.setText("Orientation Y: " + values[1]);  
 zViewO.setText("Orientation Z: " + values[2]);  
 }  
 if (sensor == SensorManager.*SENSOR\_ACCELEROMETER*) {  
 xViewA.setText("Accel X: " + values[0]);  
 yViewA.setText("Accel Y: " + values[1]);  
 zViewA.setText("Accel Z: " + values[2]);  
 }  
 }  
 }  
  
 public void onAccuracyChanged(int sensor, int accuracy) {  
 Log.*d*(tag,"onAccuracyChanged: " + sensor + ", accuracy: " + accuracy);  
 }  
 @Override  
 protected void onResume() {  
 super.onResume();  
 *// register this class as a listener for the orientation and accelerometer sensors* sm.registerListener(this,  
 SensorManager.*SENSOR\_ORIENTATION* |SensorManager.*SENSOR\_ACCELEROMETER*,  
 SensorManager.*SENSOR\_DELAY\_NORMAL*);  
 }  
  
 @Override  
 protected void onStop() {  
 *// unregister listener* sm.unregisterListener(this);  
 super.onStop();  
 }  
}  
public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 }  
}

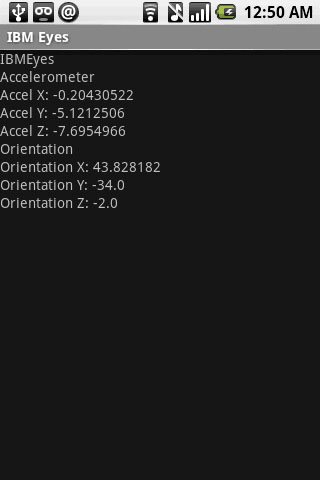


Рисунок 2.3 – Результат работы приложения, отслеживающего изменениями значений датчиков ориентации и акселерометра.

**Датчик CPS.** GPS – спутниковая система глобального позиционирования. Работа с датчиком GPS в Android отличается от работы с другими датчиками. Для геолокации в Android API предназначен пакет android .location, содержащий такие классы как Location, LocationListener и LocationManager. Специфика объясняется тем, что задача определения координат может решаться не только с помошью датчика GPS. но и другими способами, однако мы далее будем обсуждать только датчик GPS. По аналогии с обычным датчиком, источниками геолокационной информации ведает класс LocationManager, создадим его при создании приложения:

In - (LocationManager) getSystemService(LOCATION\_SERVICE);

Далее представлен пример программы, позволяющей определить координаты устройства:

public class MyLocationActivity extends Activity implements LocationListener

{

private static final String TAG=MyLocationActivity.class.getName();

@Override

public void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

LocationManager lm =

(LocationManager)getSystemService(Context.LOCATION\_SERVICE);

lm.requestLocationUpdates(LocationManager.GPS\_PROVIDER, 0, 0, this);

}

@Override

public void onLocationChanged(Location location)

{

if (location != null)

{

Log.d(TAG, "Широта="+location.getLatitude());

Log.d(TAG, "Долгота="+location.getLongitude());

}

}

@Override

public void onProviderDisabled(String provider)

{

}

@Override

public void onProviderEnabled(String provider)

{

}

@Override

public void onStatusChanged(String provider, int status, Bundle extras)

{

}

}

1. **Варианты заданий**

|  |  |
| --- | --- |
| 1 | Создать приложение, которое при нажатии на кнопку будет считывать данные с датчика света, и в зависимости от времени суток будет менять цвет фона приложения. |
| 2 | Создать приложение, которое будет подсчитывать угол наклона смартфона по одной из осей. |
| 3 | Создать приложение, которое будет менять яркость экрана при повороте смартфона. |
| 4 | Создать приложение, которое подсчитывает количество приседаний (значение счетчика меняется при движении смартфона вниз) |
| 5 | Создать приложение, которое будет выводить данные GPS на экран |
| 6 | Создать приложение, которое будет подсчитывать скорость размаха |
| 7 | Создать приложение, которое будет подсчитывать количество шагов |
| 8 | Создать приложение, которое подсчитывает скорость вращения смартфона вокруг своей оси |
| 9 | Создать приложение, которое при вращении смартфона против часовой стрелки делает экран желтого цвета, а при вращении по часовой стрелки зеленым |
| 10 | Создать приложение, которое при движении смартфона издает звуковой сигнал |

**4. Содержание отчёта**

4.1. Цель работы.

4.2. Выполненные задания по варианту.

4.3. Скриншоты по проделанной работе.

4.4. Выводы по результатам работы.

# 5. Контрольные вопросы

1. Какие датчики доступны для приложений, выполняющихся на платформе Android?

2. Как изменить систему отсчета при определении положения устройства

в пространстве?

3. Состав и назначение объектов SensorEvent, возвращаемых каждым Датчиком.

4. Как использовать вибрацию устройства в качестве физического отклика для событий внутри приложения?

5. Как использовать объект SensorManager?

6. Как отслеживать и интерпретировать показания датчиков с помощью

SensorEventListener?

7. Какой пакет и какие его классы предназначены для работы с обычными датчиками в Android API?

8. Перечислите, какие уровни точности дает объект-сенсор при вызове метода [onAccuracyChanged](http://developer.android.com/reference/android/hardware/SensorEventListener.html#onAccuracyChanged(android.hardware.Sensor,%20int)).

9. Опишите основные этапы процесса разработки практического мобильного приложения с использованием датчиков.

10. Что такое кажущееся ускорение?