

Федеральное агентство образования Российской Федерации  
Государственное образовательное учреждение  
высшего профессионального образования  
«Владимирский государственный университет»  
Кафедра ИСПИ

***Лабораторная работа***  
***по дисциплине «»***  
**«Работа с сенсорами на операционной системе Android»**

*Разработал:*

Владимир, 2015

## Цель работы

Разработать мобильное приложение, которое будет считывать сигналы сенсоров устройства и представлять их в удобном для восприятия виде.

## Теоретические сведения

Один из приятных аспектов работы с платформой Android заключается в возможности получить доступ к некоторым полезным компонентам самого устройства. До сих пор разработчиков мобильных устройств разочаровывала невозможность доступа к их внутреннему оборудованию. Хотя между вами и металлом все же остается прослойка Java-среды Android, команда разработчиков Android вывела многие возможности аппаратуры на поверхность. А так как это платформа с открытым исходным кодом, можно засучить рукава и написать собственный код для решения своих задач.

Ниже описаны некоторые аппаратно-ориентированные функции, содержащиеся в SDK Android.

Функция	Описание
android.hardware.Camera	Класс, позволяющий приложениям взаимодействовать с видеокамерой в целях фотосъемки, записи изображений с экрана предварительного просмотра или для изменения параметров настройки.
android.hardware.SensorManager	Класс, обеспечивающий доступ к внутренним датчикам платформы Android. Не каждое устройство на платформе Android поддерживает все датчики из SensorManager, однако интересно обдумать такие возможности. (Краткое описание имеющихся датчиков приведено ниже.)
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. Например, код из этой статьи содержит класс, который использует этот интерфейс для контроля ориентации устройства и показаний встроенного акселерометра.
android.media.MediaRecorder	Класс, используемый для записи

Функция	Описание
	медиафрагментов, который можно применять для записи звуков в определенном месте (например, в детской комнате). Можно также анализировать аудиофрагменты для контроля доступа в помещение и в целях безопасности. Например, можно открывать дверь собственным голосом в обычное время своего прихода, вместо того, чтобы обращаться к консьержу за ключом.
android.FaceDetector	Класс, который позволяет распознавать лицо человека по хранящейся в памяти фотографии. Ничто не удостоверяет личность лучше, чем лицо. Если использовать его для блокировки устройства, вам больше не придется запоминать пароли – достаточно биометрических возможностей мобильного телефона.
android.os.*	Пакет, содержащий несколько полезных классов для взаимодействия с операционной средой, включая управление питанием, поиск файлов, обработчик и классы для обмена сообщениями. Как и многие другие портативные устройства, телефоны на базе Android могут потреблять достаточно много электроэнергии. Обеспечение "бодрствования" устройства в нужный момент, чтобы проконтролировать нужное событие, - важный аспект проектирования, заслуживающий особого внимания.
java.util.Date java.util.Timer java.util.TimerTask	При измерении событий реального мира часто имеют значение дата и время. Например, класс java.util.Date позволяет получить метку времени, когда происходит какое-либо событие или возникает определенное состояние. java.util.Timer и java.util.TimerTask можно использовать соответственно для выполнения периодических действий по расписанию или разового действия в определенный момент времени.

Android.hardware.SensorManager содержит 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.

Центром сенсорных приложений служит интерфейс `SensorListener`. Он 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 (подробнее об этом ниже). 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.

Для взаимодействия с датчиком приложение должно зарегистрироваться на прием действий, связанных с одним или несколькими датчиками.

Регистрация осуществляется с помощью метода `registerListener` класса `SensorManager`. Пример кода для этой статьи демонстрирует, как приложение регистрируется и отменяет регистрацию с помощью `SensorListener`.

Помните, что не каждое устройство Android поддерживает тот или иной датчик, указанный в SDK. Если на конкретном устройстве тот или иной датчик отсутствует, ваше приложение должно обрабатывать эту ситуацию аккуратно.

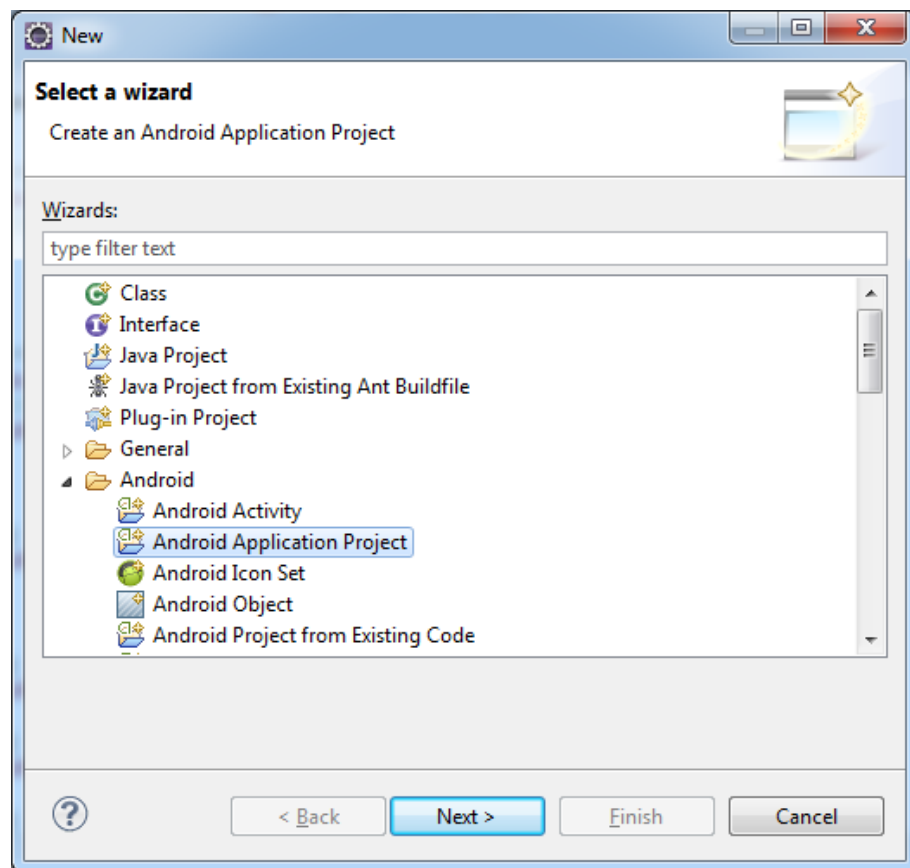
### **Порядок выполнения работы**

**Замечание:** для разработки использовалась IDE Eclipse Java EE IDE for Web Developers версии Kepler Service Release 2 с установленным плагином для разработки под Android


1. Создать новый Android Application Project;
2. Добавить на экран активности элементы для отображения значений датчиков;
3. Объявить и инициализировать переменные, связанные с элементами активности.
4. Создать обработчики событий для событий изменения значений сенсоров
5. Связать полученные обработчиком данные с переменными элементов активности
6. Выполнить индивидуальные задания
7. Оформить отчет


## Пример

Для начала создадим шаблонный проект Android приложения



New Android Application



 The prefix 'com.example.' is meant as a placeholder and should not be used

Application Name:

Project Name:


Package Name:


Minimum Required SDK:

Target SDK:


Compile With:

Theme:

 The application name is shown in the Play Store, as well as in the Manage Application list in Settings.



New Android Application



Configure Project

☒ Create custom launcher icon

☒ Create activity

☐ Mark this project as a library


☒ Create Project in Workspace

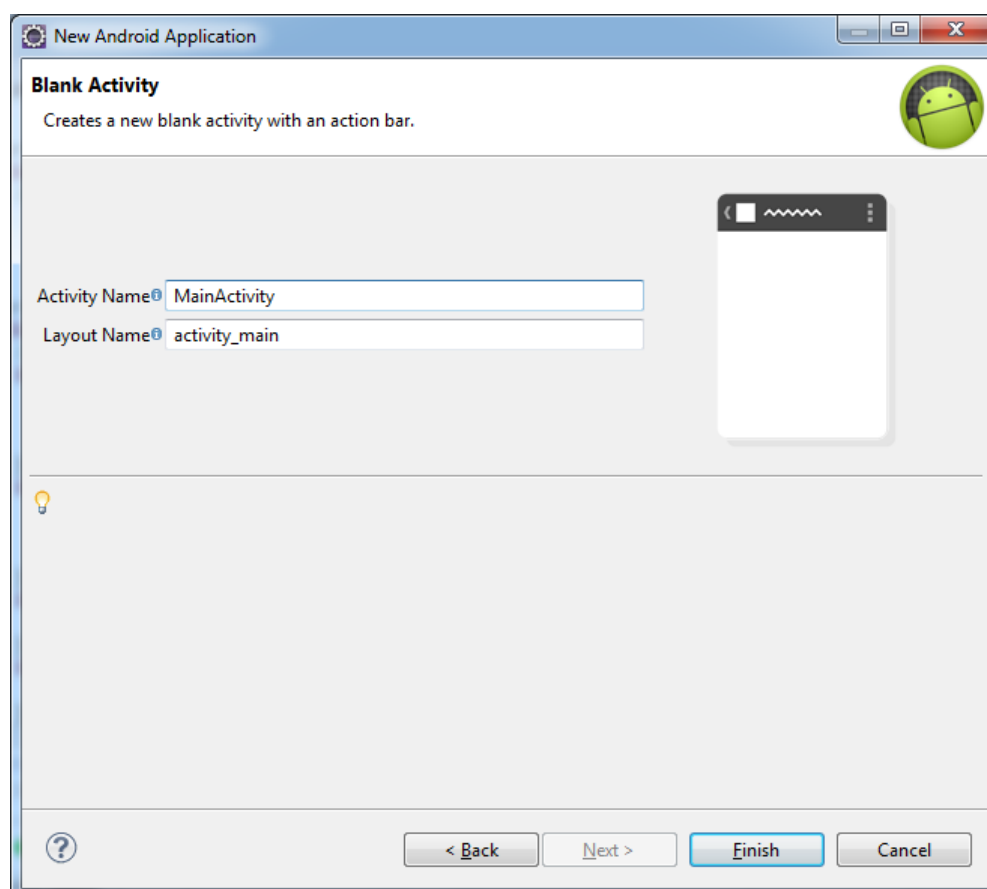
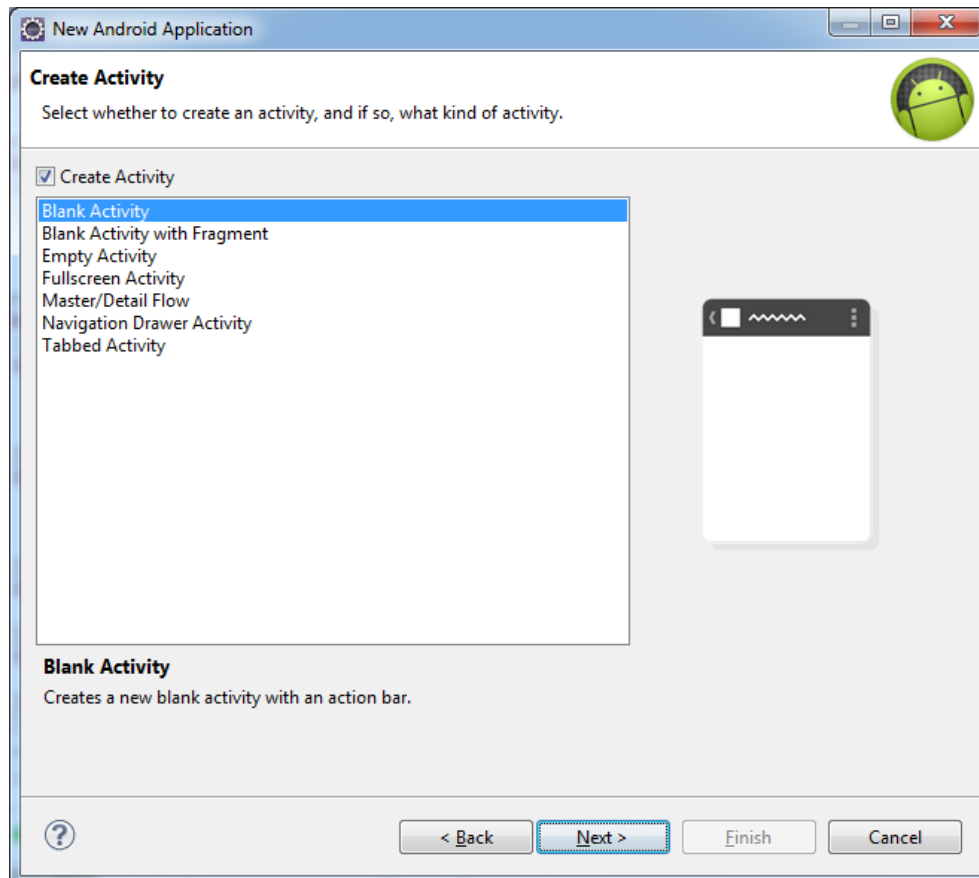
Location:

Working sets

☐ Add project to working sets

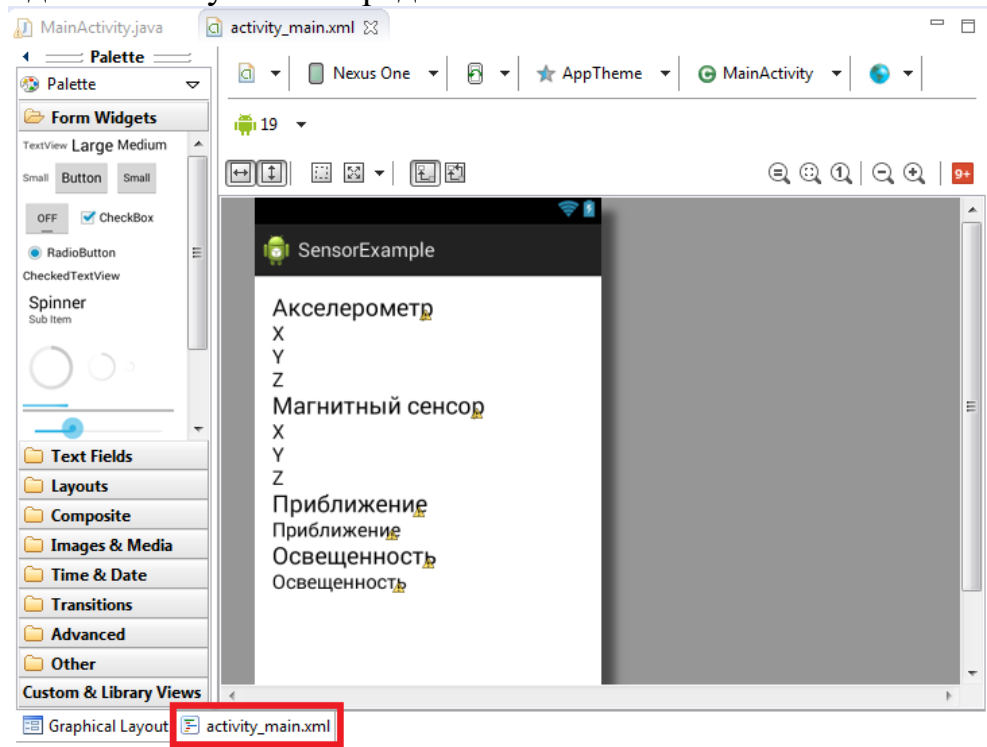
Working sets:



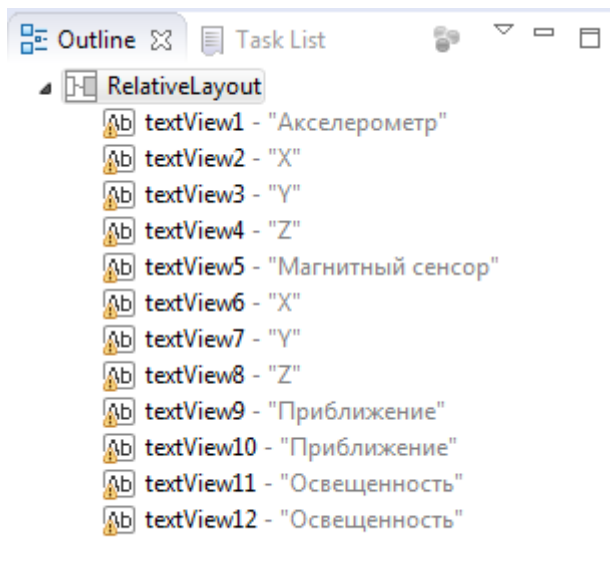


После того, как IDE создаст весь необходимый минимум файлов, можно перейти к настройке отображения элементов активности. Для этого

откройте файл `activity_main.xml` в папке `SensorExample/res/layout` и перейдите на визуальное представление активности



Для отображения использован следующий набор элементов – корневым элементом является `RelativeLayout`, на котором расположены `TextView`. На изображении, большие элементы будут использоваться как заголовки, маленькие для отображения значений. Вот как выглядит иерархия элементов:



Далее можно перейти к созданию полей класса, которые будут связаны с элементами отображения. Открываем класс `MainActivity.java` (в папке `SensorExample/src/com.example.sensorexample`) и объявляем следующие поля:

```
TextView aX;  
TextView aY;  
TextView aZ;
```



```
TextView mX;  
TextView mY;  
TextView mZ;  
TextView proximity;  
TextView light;
```

После чего, в методе onCreate данные поля привязываем к элементам на activity\_main.xml:

```
aX = (TextView)findViewById(R.id.textView2);  
aY = (TextView)findViewById(R.id.textView3);  
aZ = (TextView)findViewById(R.id.textView4);  
mX = (TextView)findViewById(R.id.textView6);  
mY = (TextView)findViewById(R.id.textView7);  
mZ = (TextView)findViewById(R.id.textView8);  
proximity = (TextView)findViewById(R.id.textView10);  
light = (TextView)findViewById(R.id.textView12);
```

Далее займемся обработчиками. Для начала сделаем так, чтобы класс MainActivity реализовывал интерфейс `SensorEventListener`

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

После чего объявим поля сенсор менеджера и самих сенсоров:

```
SensorManager sensorManager;  
Sensor aSensor;  
Sensor pSensor;  
Sensor mSensor;  
Sensor lSensor;
```

и инициализируем их в onCreate

```
sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);  
aSensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);  
mSensor = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);  
pSensor = sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);  
lSensor = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
```

Теперь можно переходить к обработке событий. Реализуем один из методов интерфейса `SensorEventListener`. Этим методом будет `onSensorChanged`.

```
@Override  
public void onSensorChanged(SensorEvent event) {  
    // TODO Auto-generated method stub  
    if(event.sensor.getType()==Sensor.TYPE_ACCELEROMETER){  
        aX.setText(Float.toString(event.values[0]));  
        aY.setText(Float.toString(event.values[1]));  
        aZ.setText(Float.toString(event.values[2]));  
    }  
    if(event.sensor.getType()==Sensor.TYPE_MAGNETIC_FIELD){  
        mX.setText(Float.toString(event.values[0]));  
        mY.setText(Float.toString(event.values[1]));  
        mZ.setText(Float.toString(event.values[2]));  
    }  
    if(event.sensor.getType()==Sensor.TYPE_PROXIMITY){  
        proximity.setText(Float.toString(event.values[0]));  
    }  
}
```

```

    }
    if(event.sensor.getType()==Sensor.TYPE_LIGHT){
        light.setText(Float.toString(event.values[0]));
    }
}

```

В данном методе мы определяем тип сенсора который вызвал событие, если это один из нужных нам сенсоров, то мы записываем его показания в соответствующие поля.

Кроме этого в методах onStart и onStop нужно сделать следующее:

```

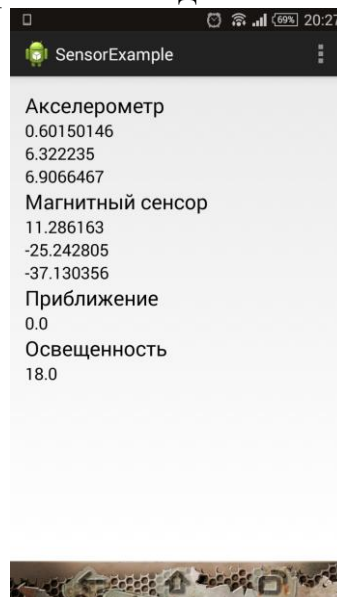
@Override
public void onStart(){
    super.onStart();
    sensorManager.registerListener(this, aSensor,
    sensorManager.SENSOR_DELAY_FASTEST);
    sensorManager.registerListener(this, mSensor,
    sensorManager.SENSOR_DELAY_FASTEST);
    sensorManager.registerListener(this, pSensor,
    sensorManager.SENSOR_DELAY_FASTEST);
    sensorManager.registerListener(this, lSensor,
    sensorManager.SENSOR_DELAY_FASTEST);
}

@Override
public void onStop(){
    super.onStop();
    sensorManager.unregisterListener(this, aSensor);
    sensorManager.unregisterListener(this, mSensor);
    sensorManager.unregisterListener(this, pSensor);
    sensorManager.unregisterListener(this, lSensor);
}

```

Это необходимо для того, чтобы ресурсы приложения не расходовались в пустую.

В итоге, после запуска приложения должно получиться следующее:



## Задания

1. Выполнить пример из лабораторной работы.

2. Выполнить одно из следующих трех заданий:
  - 2.1. Используя магнитный сенсор написать приложение, в котором можно перемещать объект на экране с помощью наклона телефона.
  - 2.2. Используя магнитный сенсор сделать компас, который бы примерно определял, в каком направлении расположен телефон (север/юг/запад/восток)
  - 2.3. Используя датчик освещенности менять яркость экрана в зависимости от уровня света в комнате.

### **Требования к оформлению отчета**

Отчет должен содержать:

1. титульный лист;
2. цель работы;
3. задание;
4. описание выполнения примера
5. описание выполнения индивидуального задания со скриншотами.
6. выводы.

### **Контрольные вопросы**

1. Какие наиболее распространенные виды датчиков используются в мобильных устройствах.
2. Приведите хотя бы по одному примеру использования для каждого датчика.
3. В каких случаях вызываются методы `onSensorChanged` и `onAccuracyChanged`.

### **Дополнительная литература**

1. <http://www.startandroid.ru/ru/uroki/vse-uroki-spiskom/287-urok-137-sensory-uskorenie-orientatsija.html>
2. <http://developer.alexanderklimov.ru/android/>