4.11. Виды сенсоров

Сайт: Samsung Innovation Campus
Курс: Мобильная разработка на Kotlin
Книга: 4.11. Виды сенсоров

Напечатано:: Murad Rezvan
Дата: понедельник, 3 июня 2024, 17:57

Оглавление

1. Датчики окружающей среды

Простейшими по обработке регистрируемых данных являются датчики окружающей среды - аппаратные датчики, каждый из которых регистрирует и передает при обращении массив из одного значения. В связи с единственностью получаемого параметра передаваемые данные не зашумлены и не нуждаются в калибровке. Чтение данных с датчика выполняется обращением к первому единственному элементу в массиве возвращаемых значений `SensorEvent.values[0]`. Всего в системе Android представлены четыре вида датчиков окружающей среды:

- датчик температуры TYPE_AMBIENT_TEMPERATURE, возвращающий значение - температуру воздуха вокруг устройства (в непосредственной близости, но не температуру самого устройства!) в градусах Цельсия;
- датчик давления TYPE_PRESSURE с возвращаемым значением атмосферного давления в ГПа вблизи от устройства;
- датчик освещённости TYPE_LIGHT, который возвращает показатель света в районе экрана устройства, измеряемый в люксах. Чаще всего используется устройствами для регулировки яркости экрана;
- датчик относительной влажности TYPE_RELATIVE_HUMIDITY, показывающий влажность воздуха в процентах.

2. Упражнение 4.11.1 Мониторинг окружающей среды

Напишем приложение, определяющее показатели окружающей среды по установленным на устройстве датчикам. В случае отсутствия датчика выводится соответственное сообщение.

1. Создайте новый проект, сделайте корневую разметку линейной вертикальной, поместите на экран 4 текстовых поля вывода под каждый тип датчика

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/temperature"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/light"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/pressure"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/humidity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
<!--здесь будет поле для точки росы-->
</LinearLayout>
```

и одно текстовое поле для вывода значения точки росы.

```
<TextView
        android:id="@+id/dewpoint"
        android:layout_width="match_parent"
        android:gravity="center"
        android:textColor="@color/design_default_color_primary"
        android:textSize="20sp"
        android:layout_height="wrap_content" />
```

Конвертируйте файл в dataBinding файл.

2. В классе `MainActivity` создайте переменную связывания данных с макетом

```
  lateinit var db: ActivityMainBinding
```

и инициализируйте её в методе `onCreate()`

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
db = DataBindingUtil.setContentView(this,R.layout.activity_main)
}
```

3. Опишите в классе текстовую константу для вывода сообщения об отсутствии датчика

```
    val noSensor = "датчик отсутствует"
```

и переменные для менеджера датчиков и отдельно под каждый датчик

```
    lateinit var sm: SensorManager
    var tSensor:Sensor? = null             //температура
    var lSensor:Sensor? = null             //свет
    var pSensor:Sensor? = null           //давление
    var hSensor:Sensor? = null           //влажность
```

В случае отсутствия датчика на устройстве инициализация соответственной переменной не произойдёт, по этому при объявлении нельзя использовать lateinit-переменные типа Sensor и необходима их null-инициализация.

4. Инициализируйте переменную менеджера в методе `onCreate()`

```
sm = getSystemService(SENSOR_SERVICE) as SensorManager
```

5. Имплементируйте классом `MainActivity` интерфейс `SensorEventListener` и добавьте в класс методы этого интерфейса.

6. В методе `onResume()` переопределите переменные для датчиков, которые установлены на устройстве и зарегистрируйте на них слушателей изменения данных.

```
override fun onResume() {
       super.onResume()
       tSensor = sm.getDefaultSensor(Sensor.TYPE_AMBIENT_TEMPERATURE)
       if(tSensor !=null)
       sm.registerListener(this,tSensor,SensorManager.SENSOR_DELAY_GAME)
       lSensor = sm.getDefaultSensor(Sensor.TYPE_LIGHT)
       if(lSensor !=null)
       sm.registerListener(this,lSensor,SensorManager.SENSOR_DELAY_GAME)
       pSensor = sm.getDefaultSensor(Sensor.TYPE_PRESSURE)
       if(pSensor !=null)
       sm.registerListener(this,pSensor,SensorManager.SENSOR_DELAY_GAME)
       hSensor = sm.getDefaultSensor(Sensor.TYPE_RELATIVE_HUMIDITY)
       if(hSensor !=null)
       sm.registerListener(this,hSensor,SensorManager.SENSOR_DELAY_GAME)
   }
```

7. Сразу же снимите регистрацию в методе `onPause()`, чтобы избежать попыток чтения датчиков неработающим приложением

```
   override fun onPause() {
       super.onPause()
       sm.unregisterListener(this) }
```

8. Запрограммируйте чтение данных с датчиков и вывод их в соответственные поля активности. При отсутствии датчика в поле выводите константу сообщения о его отсутствии.

```
   override fun onSensorChanged(event: SensorEvent?) {
       var h = 0f
       var t = 0f
       if (tSensor == null) db.temperature.text = "ТЕМПЕРАТУРА: " + noSensor
       else if (event!!.sensor.type == tSensor!!.type) {
           t = event.values[0]; db.temperature.text = "ТЕМПЕРАТУРА: " + t   }
       if (lSensor == null) db.light.text = "ОСВЕЩЁННОСТЬ: " + noSensor
       else if (event!!.sensor.type == lSensor!!.type) db.light.text =
           "ОСВЕЩЁННОСТЬ: " + event.values[0]
       if (pSensor == null) db.pressure.text = "АТМОСФЕРНОЕ ДАВЛЕНИЕ: " + noSensor
       else if (event!!.sensor.type == pSensor!!.type) db.pressure.text =
           "АТМОСФЕРНОЕ ДАВЛЕНИЕ: " + event.values[0]
       if (hSensor == null) db.humidity.text = "ОТНОСИТЕЛЬНАЯ ВЛАЖНОСТЬ: " + noSensor
       else if (event!!.sensor.type == hSensor!!.type) {
           h = event.values[0]; db.humidity.text = "ОТНОСИТЕЛЬНАЯ ВЛАЖНОСТЬ: " + h
       }
   override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) { }
```

9. Метод `onAccuracyChanged()` оставьте незаполненным.

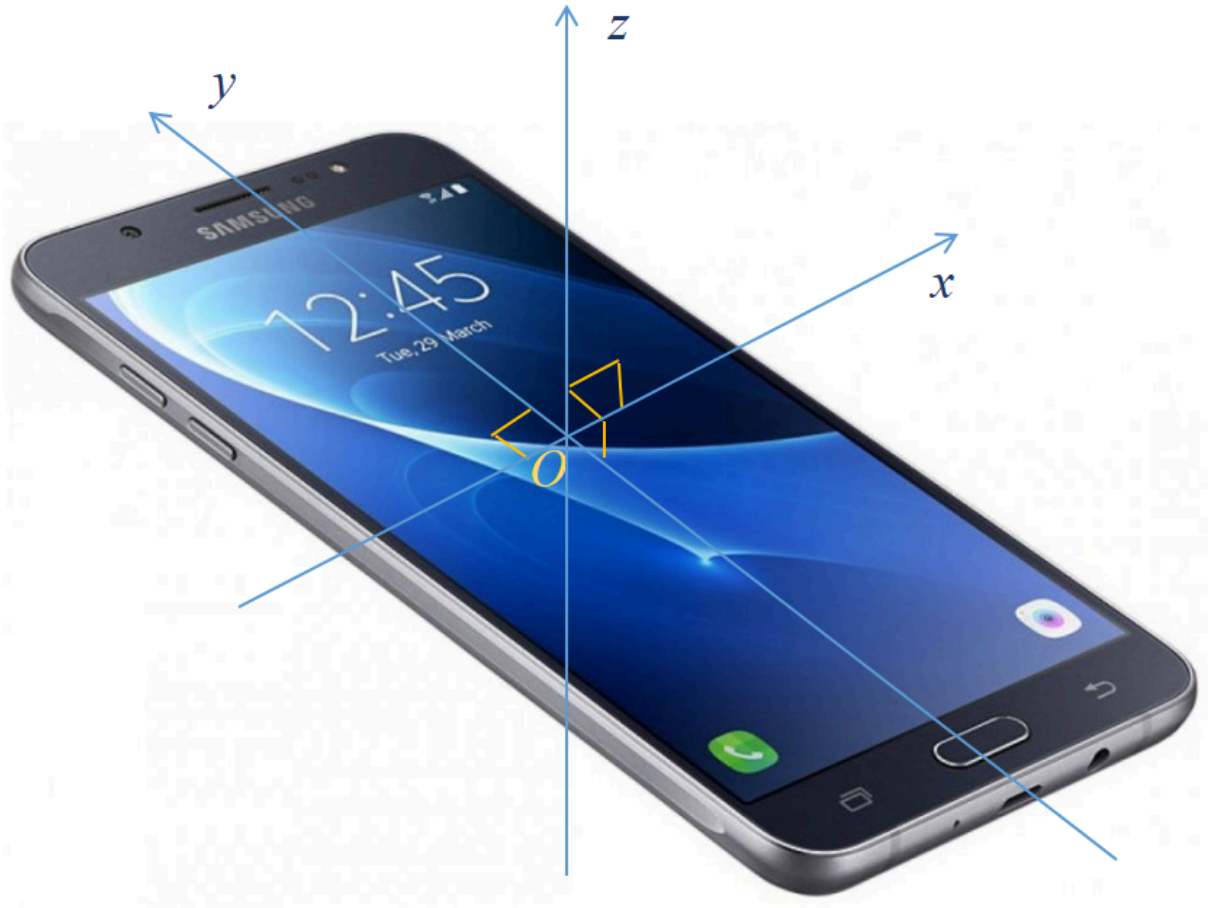
10. Реализуйте программный датчик получения точки росы на основании показаний аппаратных датчиков. Для этого в метод `onSensorChanged()` добавьте формулу для вычисления точки росы и выведите значение или сообщение о невозможности вычисления в соответственное поле экрана

```
var str = ""
        if (tSensor != null && hSensor != null) {
                var dewPoint =
                243.12 * (Math.log(h / 100.0) + 17.62 * t / (243.12 + t)) /
                (17.62 - Math.log(h / 100.0) - 17.62 * t / (243.12 + t))
                str = "точка росы: " + dewPoint}
            else str = "нет возможности вычислить точку росы"
        db.dewpoint.text = str
```

11. Запустите приложение на устройстве и проверьте показания датчиков при помещении устройства в более холодное или более тёплое место, при затемнении экрана рукой.

3. Датчики движения

Измерение положения устройства в пространстве выполняют датчики движения. Ориентация декартова пространства не привязана к экрану устройства и выполнена по принципу: экран располагается в плоскости (xOy), ось аппликат направлена снизу от задней стенки гаджета вверх к экрану, начало отсчёта размещается в центре экрана. Направления осей в плоскости (xOy) ориентируют не экран, а пространство: ось абсцисс (Ox) - горизонтальная ось с направлением "лево - право" (с запада на восток); ось ординат (Oy) - вертикальная с направлением "назад - вперёд" (с юга на север). При повороте экрана система координат остаётся зафиксирована и не поворачивается вместе с экраном.



Все современные (и не очень современные) смартфоны обязательно имеют акселерометр, всё чаще в устройства встраиваются компас и гироскоп.

Движение устройства определяется в двух измерениях:

- относительно системы отсчёта самого устройства (рис. 4.10.1);
- относительно абсолютной системы отсчёта (геомагнитная система мира).

В отличие от датчиков окружающей среды, имеющих только одно измерение, датчики движения замеряют более одного показателя. Но часто при анализе показателей сенсора требуется не разбросанный набор данных, а одно усреднённое измерение. В таком случае создаётся программный датчик, на котором получаются усреднённые данные со всех показателей. Такие датчики с меньшей точностью дают показания, но и не всегда бывает необходимость в сверхточных измерениях. Замена аппаратных сенсоров программными удешевляет стоимость устройства, при этом не сильно изменяя его функционал.

Всего в инструментарии Android SDK находятся 10 видов датчиков

- Акселерометр калиброванный TYPE_ACCELEROMETER измеряет динамическое ускорение по трём направлениям (Ox, Oy, Oz) в базовых единицах измерения СИ. Определяет вибрацию, поворот устройства, ускорение изменения координат. Используется при определении ориентации экрана устройства, в управлении наклонами устройства.

Измеряемые показатели:

SensorEvent.values[0] - ускорение по оси Ox

SensorEvent.values[1] - ускорение по оси Oy

SensorEvent.values[2] -ускорение по оси Oz

- Акселерометр некалиброванный TYPE_ACCELEROMETER_UNCALIBRATED измеряет динамическое ускорение по трём направлениям (Ox, Oy, Oz) в базовых единицах измерения СИ без калибровки. Определяет вибрацию, поворот устройства, ускорение изменения координат

Получаемые значения:

SensorEvent.values[0] - фактическое ускорение по оси Ox без калибровки

SensorEvent.values[1] - фактическое ускорение по оси Oy без калибровки

SensorEvent.values[2] - фактическое ускорение по оси Oz без калибровки

SensorEvent.values[3] - фактическое ускорение по оси Ox с учётом стандартного смещения

SensorEvent.values[4] - фактическое ускорение по оси Oy с учётом стандартного смещения

SensorEvent.values[5] - фактическое ускорение по оси Oz с учётом стандартного смещения

- Датчик гравитации TYPE_GRAVITY измеряет силу тяжести телефона в базовых единицах измерения СИ. Используется в совокупности с акселерометром для управления приложениями при помощи наклонов устройства.

Измеряемые показания:

SensorEvent.values[0] - сила тяжести, направленная по оси Ox

SensorEvent.values1] - сила тяжести по направлению оси Oy

SensorEvent.values2] - сила тяжести, направленная по оси Oz

- Гироскоп TYPE_GYROSCOPE измеряет угол вращения устройства вдоль координатных осей. Данный датчик является калиброванным, то есть даёт абсолютные значения вращений.

Измеряемые показатели:

SensorEvent.values[0] - угол поворота вокруг оси Ox

SensorEvent.values[1] - угол поворота вокруг оси Oy

SensorEvent.values[2] - угол поворота вокруг оси Oz

- Гироскоп некалиброванный TYPE_GYROSCOPE_UNCALIBRATED снимает показатели вращения в калиброванном и некалиброванном виде

Измеряемые показатели:

SensorEvent.values[0] - фактический угол поворота вокруг оси Ox

SensorEvent.values[1] - фактический угол поворота вокруг оси Oy

SensorEvent.values[2] - фактический угол поворота вокруг оси Oz

SensorEvent.values[3] - угол поворота вокруг оси Ox, вычисленный на основании нескольких измерений

SensorEvent.values[4] - рассчитанный угол поворота вокруг оси Oy, вычисленный на основании нескольких измерений

SensorEvent.values[5] - угол поворота вокруг оси Oz, вычисленный на основании нескольких измерений

- Акселератор TYPE_LINEAR_ACCELERATION подобен акселерометру, но измеряет ускорения без учёта силы тяжести в единицах измерения СИ

Измеряемые показатели:

SensorEvent.values[0] - сила ускорения вдоль оси Ox без учёта гравитации

SensorEvent.values[1] - сила ускорения вдоль оси Oy без учёта гравитации

SensorEvent.values[2] - сила ускорения вдоль оси Oz без учёта гравитации

- Датчик вращения TYPE_ROTATION_VECTOR с помощью гироскопа измеряет проекции векторов поворота устройства в пространстве на угол по каждой из осей относительно магнитных полюсов Земли. При этом ось Oy показывает строго на север. Сам вектор вращения

представляется тремя измерениями и углом поворота вдоль измеряемой оси (x * sin(a/2), y * sin(a/2), z * sin(a/2)), где x, y, z - величины, измеряемые акселерометром и a - угол поворота вектора.

Измеряемые показатели:

SensorEvent.values[0] - проекция вектора вращения на ось Ox, вычисляется как x*sin(a/2)

SensorEvent.values[1] - проекция вектора вращения на ось Oy, вычисляется как y*sin(a/2)

SensorEvent.values[2] - проекция вектора вращения на ось Oz, вычисляется как z*sin(a/2)

С версии SDK 18 в измерения добавлены два показателя:

обязательный SensorEvent.values[3] - скалярная мера угла поворота cos(a/2)

и опциональный SensorEvent.values[4] - рассчитываемая точность направления вектора (измеряется в радианах)

- Датчик выполненных шагов TYPE_STEP_COUNTER считает количество шагов пользователя и сбрасывается при перезагрузке устройства. При отмене регистрации датчик перестаёт вести подсчёт шагов, то есть для постоянного учёта пройденных шагов рекомендуется не отменять регистрацию данного сенсора. Для использования датчика шагов в приложении необходимо в файле манифеста дать разрешение на его использование `android.permission.ACTIVITY_RECOGNITION`.

Измеряемый показатель:

SensorEvent.values[0] - общее количество шагов, выполненных от момента загрузки системы.

- С API 19 в систему был добавлен датчик единичного шага TYPE_STEP_DETECTOR, выполняющий регистрацию каждого шага. Датчик не производит подсчёта количества выполненных шагов, он определяет момент времени, когда изменилось ускорение, то есть нога коснулась поверхности. Датчик всегда в качестве значения SensorEvent.values[0] возвращает константу 1.0. Для доступа приложения к показателю датчика требуется разрешение в манифесте `android.permission.ACTIVITY_RECOGNITION`.

- Виртуальный датчик значительного движения TYPE_SIGNIFICANT_MOTION, основанный на показаниях акселерометра, был добавлен в SDK с версии 18. Данный датчик пробуждающий, срабатывает при значительном изменении показателя акселерометра и отключается самостоятельно. При этом не создаётся массива данных SensorEvent, но генерируется объект TriggerEvent с двумя показателями: время происхождения события (в наносекундах) и значения показателей в зависимости от используемых физических датчиков.

4. Упражнение 4.11.2 Использование датчиков перемещения

Практически все современные гаджеты имеют датчик гироскоп. В проект упражнения 4.11.1 добавим чтение гироскопа.

1. В ресурсы проекта в папку drawable добавьте картинку, которую будете вращать.

2. На главной активности добавьте в разметку ImageView, разместите в него добавленную картинку

```
<ImageView
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:layout_gravity="center"
        android:id="@+id/droid"
        android:src="@drawable/droid"/>
```

3. Объявите в классе ещё одну переменную типа Sensor

```
var rvSensor: Sensor? = null
```

4. В методе `onResume()` найдите гироскоп, используемый по умолчанию, и задайте ему слушателя с самой быстрой частотой обновления

```
rvSensor = sm.getDefaultSensor(Sensor.TYPE_GYROSCOPE)
        if(rvSensor !=null)
        sm.registerListener(this,rvSensor, SensorManager.SENSOR_DELAY_FASTEST)
```

Поскольку в методе `onPause()` отменяются все слушатели, то новый датчик не будет читаться в режиме паузы.

5. В методе `onSensorChanged()` выполните проверку на наличие датчика, прочитайте с него смещение по оси Oz (values[2]) и задайте ImageView поворот на это смещение. Поскольку показатель измеряется в достаточно малом диапазоне, то задайте кратность углу вращения умножением на 10.

```
if (event!!.sensor.type == rvSensor!!.type){
            db.droid.setRotation (event.values[2]*10)}
```

6. Запустите приложение и выполните вращение устройства вокруг оси Ox или Oy. Для того, чтобы изображение на активности стало вращаться, поворачивайте устройство вокруг оси Oz.

   Для вращения изображения по всем трём осям нужно задать матрицу вращения на основании показаний сенсора TYPE_ROTATION_VECTOR или использовать объявленный устаревшим датчик TYPE_OPIENTATION.

7. Замените значение поворота картинки на `event.values[0]` и `event.values[1]` и посмотрите на изменения, происходящие в приложении.

Поменяйте тип используемого сенсора на некалиброванный гироскоп или датчик вращения и тоже посмотрите изменения в работе приложения.

5. Датчики положения

Наряду с датчиками движения, в аппаратном составе устройства присутствуют датчики положения. Эти датчики не реагируют на такие действия с устройством, как наклоны, повороты, перенос. они просто фиксируют расположение устройства в пространстве относительно полюса Земли.

Чаще всего датчики положения используются в совокупности с акселерометром. Именно по этой причине акселерометр считается не только датчиком движения, но и датчиком положения.

Кроме акселерометра, в состав SDK системы Android входят следующие типы датчиков:

- Магнитометры калиброванный и некалиброванный TYPE_MAGNETIC_FIELD / TYPE_MAGNETIC_FIELD_UNCALIBRATED измеряют магнитные поля (в микроТесла) в направлениях по трём координатным осям. Некалиброванный датчик кроме трёх абсолютных измерений также возвращает величину магнитного смещения по каждой из осей. В случае некалиброванного датчика предполагается, что температурная компенсация показателя магнитного поля не производится.

Измеряемые показатели

в зависимости от калибровки:

SensorEvent.values[0] - геомагнитное поле по оси Ox (калиброванное / некалиброванное)

SensorEvent.values[1] - геомагнитное поле по оси Oy (калиброванное / некалиброванное)

SensorEvent.values[2] - геомагнитное поле по оси Oz (калиброванное / некалиброванное)

для некалиброванного типа датчика считывается влияние металлических компонентов устройства:

SensorEvent.values[3] - калибровочная поправка по оси Ox

SensorEvent.values[4] - калибровочная поправка по оси Oy

SensorEvent.values[5] -калибровочная поправка по оси Oz

- Датчики поворота аналогичны датчику движения TYPE_ROTATION_VECTOR и имеют тот же массив измерений

SensorEvent.values[0] - проекция вектора вращения на ось Ox, вычисляется как x*sin(a/2)

SensorEvent.values[1] - проекция вектора вращения на ось Oy, вычисляется как y*sin(a/2)

SensorEvent.values[2] - проекция вектора вращения на ось Oz, вычисляется как z*sin(a/2)

При этом

TYPE_GAME_ROTATION_VECTOR не использует геомагнитное поле для получения показателей. Таким образом, датчик показывает абсолютный разворот устройства без привязки к сторонам света. Со временем точка отсчёта может быть изменена.

TYPE_GEOMAGNETIC_ROTATION_VECTOR для ориентации вместо гироскопа использует магнитометр. Это позволяет экономить расход батареи, но накладывает ограничения на точность измерений за счёт чувствительности магнитометра к окружающим металлическим предметам. Лучше всего данный датчик будет работать в "чистом поле" без следов урбанизации.

- Датчик приближения TYPE_PROXIMITY определяет расстояние устройства от различных поверхностей (в сантиметрах). Выводит девайс из режима ожидания и сна, то есть является пробуждающим датчиком. Именно этот датчик отключает экран смартфона при поднесении его к уху во время разговора.

Измеряемые показатели:

SensorEvent.values[0] - расстояние до датчика.

> Не смотря на объявление устаревшим датчика ориентации TYPE_ORIENTATION, некоторые вендоры по-прежнему его включают в состав своих устройств. Однако, использование датчика в классе SensorManager заменено на метод getOrientation()

6. Упражнение 4.10.3 Использование датчика приближения

Дополним проект, разработанный в упражнениях 4.11.1 и 4.11.2 чтением датчика приближения.

1. Объявите в классе ещё одну переменную - сенсор

```
var prSensor: Sensor? = null //приближение
```

2. В методе `onResume()` найдите датчик приближения и зарегистрируйте на нём слушателя

```
prSensor = sm.getDefaultSensor(Sensor.TYPE_PROXIMITY)
        if(prSensor != null)
        sm.registerListener(this,prSensor,SensorManager.SENSOR_DELAY_GAME)
```

3. В методе `onSensorChanged()` слушателя добавьте проверку на датчик приближения и определите для него действия

```
if(event!!.sensor.type == prSensor!!.type )  {

        }
```

4. При приближении к устройству какого-либо предмета ближе, чем на 2 см, будем подавать звуковой сигнал

```
if(event!!.sensor.type == prSensor!!.type &&  //работает датчик приближения
event.values[0] < 2)  {                                  // ближе 2 см находится некоторый предмет
            val alert = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION) //используем системный звук уведомлений
            val mp = MediaPlayer.create(applicationContext, alert)   // создаём звуковой объект
            mp.start()                                                                                  // и запускаем его
        }
```

5. Запустите приложение и поднесите руку в экрану ближе, чем на 2 см.

Часто для определения расстояния используют значение максимально допустимого расстояния `maximumRange` у датчика

```
event.values[0] < prSensor!!.maximumRange
```

Готовое приложение по трём упражнениям можно посмотреть здесь.

Начать тур для пользователя на этой странице