

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Брянский государственный технический университет

**Утверждаю**

**Ректор университета**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.Н.Федонин**

**«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2017 г.**

**Программирование МОБИЛЬНЫХ СИСТЕМ**

**ОБРАБОТКА ДАННЫХ С ДАТЧИКОВ МОБИЛЬНЫХ УСТРОЙСТВ**

**Методические указания**

**к выполнению лабораторной работы №6**

**для студентов очной формы обучения по направлениям подготовки 02.03.03 – «Математическое обеспечение и администрирование информационных систем», 09.03.01 – «Информатика и вычислительная техника» и 09.03.04 – «Программная инженерия»**

**Брянск 2017**

УКД 004.43

Программирование мобильных систем. Обработка данных с датчиков мобильных устройств. [Текст] + [Электронный ресурс]: Методические указания к выполнению лабораторной работы №6 для студентов очной формы обучения по направлениям подготовки 02.03.03 – «Математическое обеспечение и администрирование информационных систем», 09.03.01 – «Информатика и вычислительная техника» и 09.03.04 – «Программная инженерия». – Брянск: БГТУ, 2017. – 11 с.

Разработал:

Д.Н.Панус

ст.преп.

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ (протокол №2 от 16.09.2016)

Научный редактор Д.А.Коростелев

Редактор издательства Л.И.Афонина

Компьютерный набор Д.Н.Панус

Темплан 2017 г., п.273

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подписано в печать 23.09.17. Формат 60х84 1/16 Бумага офсетная. Офсетная печать. Усл.печ.л. 1,8. Уч.-изд.л. 1,8 Тираж 1 экз. Заказ Бесплатно.

Издательство Брянского государственного технического университета

241035, Брянск, бульвар 50-летия Октября, 7, БГТУ. 58-82-49.

Лаборатория оперативной полиграфии БГТУ, ул. Институтская, 16.

**Цель работы**

Целью работы является изучение особенности функционирования датчиков в устройствах на базе ОС Android.

Продолжительность работы – 5 часов.

**Порядок выполнения работы**

1. Произвести выбор выполняемого варианта лабораторной работы на основе своего порядкового номера в группе по алфавиту.
2. Выяснить возможности предоставляемые тестовой средой для демонстрации результата работы(телефон или эмулятор ОС Android).
3. Ознакомиться с теоретическим материалом для лабораторной работы. Разобраться с теорией хранения пользовательской и системной информации приложениями в ОС Android.
4. Проанализировать какие компоненты, используемые в ОС Android необходимо применить в данной лабораторной работе.
5. Ознакомиться с особенностями интерпретации данных получаемых с датчиков и компонентов тестовой среды.
6. На примере возможностей тестового окружения произвести анализ физических и виртуальных датчиков, и способов от отличия в приложении.
7. Произвести проектирование и реализацию интерфейса приложения.
8. Произвести проектирование и реализацию средств хранения информации приложением.
9. Описать в виде XML требуемые ресурсы приложения.
10. Реализовать в приложении общие требования к лабораторной работе.
11. Реализовать специфичные для варианта требования в лабораторной работе, ознакомившись, mc правилами их реализации.
12. Произвести сборку готового приложения в виде \*.apk файла приложения.
13. Продемонстрировать полученный результат преподавателю.

**Использование датчиков и объекта SensorManager**

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.

Класс *SensorManager* разработан для работы с аппаратными датчиками, доступными в устройствах под управлением Android. Используйте метод *getSystemService*, чтобы получить ссылку на объект *SensorManager*, как показано в следующем фрагменте

|  |
| --- |
| String service\_name = Context.SENSOR\_SERVICE; SensorManager sensorManager = (SensorManager)getSystemService(service\_ name); |

Объекты *Sensor* позволяют не взаимодействовать с аппаратным обеспечением датчиков напрямую, они описывают свойства представляемого датчика, включая его тип, название, имя производителя, а также подробности о его точности и диапазоне.

Данный класс содержит набор констант, используемых для определения, какой именно тип датчика представляет объект *Sensor*. Эти константы имеют вид *Sensor.TYPE\_<TYPE>*.

**Датчики поддерживаемые платформой Android**

Ниже представлены типы датчиков, доступные на сегодняшний день; имейте в виду, что набор датчиков, действительно доступных для вашего приложения, определяется аппаратной начинкой устройства.

* **Sensor.TYPE\_ACCELEROMETER.** Трех осевой акселерометр, возвращающий текущее ускорение по трем осям в м/с.
* **Sensor.TYPE\_GYROSCOPE.** Гироскопт (гироскопический датчик), возвращающий текущее положение устройства в пространстве в градусах по трем осям.
* **Sensor.TYPE\_LIGHT.** Датчик окружающей освещенности, возвращающий одиночное значение, которое описывает внешнюю освещенность в люксах. Этот тип датчиков обычно используется для динамического изменения яркости экрана.
* **Sensor.TYPE\_MAGNETIC\_FIELD.** Датчик магнитного поля, определяющий текущие показатели магнитного поля в микротеслах по трем осям.
* **Sensor.TYPE\_ORIENTATION.** Датчик ориентации, возвращающий положение устройства в градусах по трем осям.
* **Sensor.TYPE\_PRESSURE.** Датчик давления, возвращающий одиночное значение — текущее давление в килопаскалях, оказываемое на устройство.
* **Sensor.TYPE\_PROXIMITY.** Датчик приближенности, который сигнализирует о расстоянии между устройством и целевым объектом (в метрах). Каким образом выбирается объект и какие расстояния поддерживаются, зависит от аппаратной реализации данного датчика. Типичное его применение — обнаружение расстояния между устройством и ухом пользователя для автоматического регулирования яркости экрана или выполнения голосовой команды.
* **Sensor.TYPE\_TEMPERATURE.** Термометр, возвращающий температуру в градусах Цельсия. В зависимости от реализации это может быть температура в помещении, удаленного датчика, показатель нагрева батареи.

**Знакомство с виртуальными датчиками**

Датчики в Android обычно работают независимо друг от друга, сигнализируя о результатах, полученных от конкретных аппаратных устройств без фильтрации или нормализации значений.

Но иногда стоит воспользоваться виртуальными датчиками, которые представляют у прощенные, уточнённые или комбинированные показания в том виде, в котором их легче использовать в некоторых приложениях.

**TYPE\_GRAVITY, TYPE\_LINEAR\_ACCELERATION** и **TYPE\_ROTATION\_VECTOR** – это пример виртуальных датчиков, которые предоставляются фреймворком. Вместо показаний какого-то определенного устройства они могут использовать комбинацию из нескольких устройств, например, акселерометра, магнитометра и гироскопа.

**Поиск датчиков**

Устройство на базе Android может включать в себя несколько реализаций одного и того же типа датчиков. Чтобы найти реализацию, использующуюся по умолчанию, вызовите метод *getDefaultSensor* из объекта *SensorManager*, передавая ему в качестве параметра тип датчика в виде одной из констант, описанных в предыдущем разделе.

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

|  |
| --- |
| Sensor defaultGyroscope = sensorManager.getDefaultSensor(Sensor.TYPE\_); |

Вы можете использовать метод *getSensorList* для получения списка всех 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); |

Чтобы получить все датчики, доступные на устройстве, используйте метод *getSensorList*, передавая ему константу *Sensor.TYPE\_ALL*, как показано ниже:

|  |
| --- |
| List<Sensor> allSensors = sensorManager.getSensorList(Sensor.TYPE\_ALL); |

**Использование датчиков**

В листинге показан стандартный способ отслеживания результатов, передаваемых аппаратными сенсорами. Реализуйте интерфейс *SensorEventListener*. Используйте *onSensorChanged* для отслеживания значений, возвращаемых датчиком, и *onAccuracyChanged* для реагирования на изменения в их точности.

|  |
| --- |
| **final** SensorEventListener mySensorEventListener = **new** SensorEventListener(){ **public void** onSensorChanged(SensorEvent sensorEvent) { *//* ***TODO Отслеживать изменения в показаниях датчиков.***} **public void** onAccuracyChanged(Sensor sensor, **int** accuracy) { *//* ***TODO Реакция на изменения в точности значений, выдаваемых датчиками.***} }; |

Параметр *SensorEvent*, который передается в метод *onSensorChanged*, содержит четыре свойства, описывающих событие, связанное с датчиком.

* **sensor**. Объект Sensor, инициировавший событие.
* **accuracy**. Точность датчика во время поступления события (низкая, средняя, высокая или недостоверная, о чем пойдет речь в следующем маркированном списке).
* **values**. Массив чисел типа float, содержащий новые полученные значения. В следующем разделе будут рассмотрены значения, возвращаемые для каждого типа датчика.
* **timestamp**. Время возникновения события, связанного с датчиком (в наносекундах).

Также можно отслеживать точность каждого отдельного датчика, используя метод *onAccuracyChanged*. Оба этих обработчика используют значение *accuracy* для представления степени точности датчика с помощью одной 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. 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 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.

**Пример управления вибрацией устройства**

Чтобы управлять вибрацией устройства, ваше приложение должно иметь полномочие *VIBRATE*. Добавьте его в манифест, как показано в следующем фрагменте кода:

|  |
| --- |
| <uses-permission android:name=**"android.permission.VIBRATE"**/> |

Управляет вибрацией устройства Сервис Vibrator, доступ к которому можно получить через метод *getSystemService*:

|  |
| --- |
| *String vibratorService = Context.VIBRATOR\_SERVICE;  Vibrator vibrator = (Vibrator)getSystemService(vibratorService);* |

Вызовите метод *vibrate*, чтобы заставить устройство вибрировать; вы можете передать в качестве параметров продолжительность вибрации или шаблон ее воспроизведения (последовательность вибрация/пауза), а также необязательный индекс, указывающий на позицию, с которой следует начать повторение этого шаблона.

|  |
| --- |
| **long**[] pattern = {1000, 2000, 4000, 8000, 16000 }; vibrator.vibrate(pattern, 0); *// Выполнить шаблон воспроизведения вибрации.* vibrator.vibrate(1000); *// Вибрировать на протяжении одной секунды.* |

Чтобы отменить вибрацию, вызовите метод *cancel*. При выходе из приложения все вибрации, инициированные им, будут отменены.

**Задание на лабораторную работу**

Необходимо создать Android приложение, способное получать информацию с доступных на устройстве датчиков и сенсоров, со следующими требованиями.

Общие требования для выполнения лабораторной работы:

1. Получить и отобразить список всех доступных на устройстве датчиков в виде списка. Элементы списка должны открывать экран с описанием датчика (производитель, точность, название и т.п.), в отдельной Активности.
2. Реализовать обработчики событий датчиков. Количество контролируемых датчиков определяется вариантом задания.
3. Для управления датчиком реализовать экран управления. На данном экране реализовать интерфейс управления частотой обновления показателей датчика.
4. Результаты работы датчиков отображаются на отдельных экранах.
5. Данные с датчиков должны предоставляться пользователю в виде понятном для восприятия.
6. Дополнительной реакцией на нажатие кнопок на экране должна быть вибрация.

**Типы заданий**

1. Реализовать экран отображающие результаты работы компаса, в текстовом виде (должно быть указано направление в текстовом виде, с не менее чем восемью направлениями, а также их числовой эквивалент)
2. Реализовать экран отображающий результаты работы компаса, в виде графического объекта отображающего направление.
3. Реализовать экран отображающий результаты работы гироскопа. Значения необходимо выводить в нормализованном виде, с максимальным пояснением отображаемых результатов.
4. Реализовать экран отображающий результаты работы гироскопа. На данном экране необходимо произвести отображение 3-х мерного объекта, который будет реализовывать поведение, аналогичное поведению реального гироскопа, для демонстрации работы.
5. Реализовать экран отображающий результаты работы датчика магнитного поля.
6. Реализовать экран отображающий результаты работы датчика температуры (при наличии датчика внешней среды, или датчика температуры аккумулятора).
7. Реализовать экран отображающий результаты работы датчика освещенности.

Комбинации заданий по вариантам:

1. 1,8
2. 2,5
3. 3,5
4. 4
5. 4,7
6. 2,3
7. 6,7
8. 1,7
9. 2,3
10. 1,5
11. 1,3
12. 2,7
13. 3,7

Итог лабораторной работы - \*.apk файл приложения, способный работать на устройстве с Android (эмулятор, при отсутствии возможности работы с физическим устройством) с версией API от 15 до 18, с целевой версией API 15 (Android 4.0.3)

**Контрольные вопросы**

1. Класс SensorManager. Зачем он необходим?
2. Перечислите основные датчики имеющиеся в android устройствах.
3. Поиск датчиков на устройстве.
4. Логика работы датчиками.
5. Какие характеристики датчиков нам доступны для принятия решения об их использовании.
6. С какой периодичностью можно получать данные с датчиков? Назовите основные именованные константы.
7. Реальные и виртуальные датчики в чем разница?
8. Функция вибрации. Назовите два основных варианта настройки вибрации по длительности.

**Список рекомендуемой литературы**

1. Майер, Р. Android 4. Программирование приложений для планшетных компьютеров и смартфонов. / Пер. с англ. – М.: Эксмо, 2013. – 816 С.
2. Коматинени, С. Android 4 для профессионалов. Создание приложений для планшетных компьютеров и смартфонов./ Коматинени С., Маклин Д. / Пер. с англ. – М.: Вильямс, 2012. – 880 С.