Лабораторная работа № 9 по теме:
«Работа с сенсорами Android-устройства»
Теоретическая часть

Большинство Android-устройств имеют встроенные сенсоры (датчики). Эти датчики способны предоставлять необработанные данные с высокой точностью и полезны, если требуется отслеживать движение в пространстве, позиционирование устройства или изменения в окружающей среде рядом с устройством. Например, игра может отслеживать показания датчика силы тяжести устройства, чтобы определять сложные жесты и движения пользователя, такие как наклон, встряхивание или вращение устройства. Погодное приложение может использовать датчик температуры и датчик влажности. Приложение для путешествий может использовать датчик геомагнитного поля и акселерометр для определения направления по компасу.

Платформа Android поддерживает три типа сенсоров:

Датчики движения измеряют силы ускорения и силы вращения по трем осям. В эту категорию входят акселерометры, датчики силы тяжести, гироскопы и датчики вектора вращения.

Датчики окружающей среды измеряют различные параметры окружающей среды, такие как температуру и давление окружающего воздуха, освещенность и влажность. В эту категорию входят барометры, фотометры и термометры.

Датчики положения измеряют физическое положение устройства. В эту категорию входят датчики ориентации и магнитометры.

Платформа Android предоставляет несколько классов и интерфейсов, которые помогают выполнять широкий спектр задач, связанных с датчиками, например:

- Определять, какие датчики доступны на устройстве.

- Определять возможности отдельного датчика, такие как его максимальный диапазон, производитель, требования к питанию и разрешение.
- Получать необработанные данные датчика и определять минимальную скорость сбора данных датчика.
- Регистрировать и удалять слушатели событий датчиков, которые отслеживают изменения датчиков.

Платформа Android позволяет получить доступ ко многим типам датчиков. Часть из них — аппаратные, часть — программные. **Аппаратные датчики** — это физические компоненты, встроенные в мобильное устройство. Они получают свои данные путем прямого измерения определенных свойств окружающей среды, таких как ускорение или напряженность геомагнитного поля.

Программные датчики не являются физическими устройствами, хотя они имитируют аппаратные датчики. Программные датчики получают данные от одного или нескольких аппаратных датчиков и иногда называются виртуальными датчиками или синтетическими датчиками. Датчик линейного ускорения и датчик силы тяжести являются примерами программных датчиков. В таблице 1 приведены датчики, поддерживаемые платформой Android.

Таблица 1. Датчики, поддерживаемые платформой Android.

Датчик	**Название**	**Тип**	**Назначение**
TYPE_ACCELEROMETER	Акселерометр (датчик ускорения)	Аппаратный	Обнаружение движения (встряхивание, наклон и т.д.)
TYPE_AMBIENT_ TEMPERATURE	Датчик температуры среды	Аппаратный	Контроль температуры воздуха
TYPE_GRAVITY	Датчик силы тяжести	Программный или аппаратный	Обнаружение движения (встряхивание, наклон и т.д.)

TYPE_GYROSCOPE	Гироскоп	Аппаратный	Обнаружение вращения (вращение, поворот и т.д.)
TYPE_LIGHT	Датчик освещенности	Аппаратный	Регулировка яркости экрана
TYPE_LINEAR_ ACCELERATION	Датчик линейного ускорения	Программный или аппаратный	Контроль ускорения по одной оси
TYPE_MAGNETIC_FIELD	Датчик геомагнитного поля	Аппаратный	Создание компаса
TYPE_ORIENTATION	Датчик ориентации	Программный	Определение положения устройства
TYPE_PRESSURE	Датчик давления	Аппаратный	Мониторинг изменения атмосферного давления
TYPE_PROXIMITY	Датчик близости	Аппаратный	Определение положения телефона во время разговора
TYPE_RELATIVE_ HUMIDITY	Датчик влажности	Аппаратный	Контроль точки росы, абсолютной и относительной влажности
TYPE_ROTATION_ VECTOR	Датчик вращения	Программный или аппаратный	Обнаружение движения и обнаружение вращения
TYPE_TEMPERATURE	Датчик температуры	Аппаратный	Контроль температуры

В таблице 2 приведено более подробное описание показания этих датчиков.

Таблица 2. Описание показаний датчиков.

Датчик	**Описание показаний**
TYPE_ACCELEROMETER	Измеряет силу ускорения в м/с^2, приложенную к устройству по всем трем физическим осям (x, y и z), включая силу тяжести.
TYPE_AMBIENT_ TEMPERATURE	Измеряет температуру окружающей среды в градусах Цельсия (°C).
TYPE_GRAVITY	Измеряет силу тяжести в м/с^2, приложенную к устройству по всем трем физическим осям (x, y, z).
TYPE_GYROSCOPE	Измеряет скорость вращения устройства в рад/с вокруг каждой из трех физических осей (x, y и z).

TYPE_LIGHT	Измеряет уровень внешней освещенности в люксах.
TYPE_LINEAR_ ACCELERATION	Измеряет силу ускорения в м/с^2, приложенную к устройству по всем трем физическим осям (x, y и z), за исключением силы тяжести.
TYPE_MAGNETIC_FIELD	Измеряет окружающее геомагнитное поле для всех трех физических осей (x, y, z) в мкТл.
TYPE_ORIENTATION	Измеряет градусы поворота устройства вокруг всех трех физических осей (x, y, z). Начиная с уровня API 3, можно получить матрицу наклона и матрицу вращения для устройства, используя датчик силы тяжести и датчик геомагнитного поля в сочетании с методом getRotationMatrix ().
TYPE_PRESSURE	Измеряет давление окружающего воздуха в гПа или мбар.
TYPE_PROXIMITY	Измеряет близость объекта в см относительно экрана устройства. Этот датчик обычно используется для определения того, подносится ли телефонная трубка к уху человека.
TYPE_RELATIVE_ HUMIDITY	Измеряет относительную влажность окружающей среды в процентах (%).
TYPE_ROTATION_ VECTOR	Измеряет ориентацию устройства, предоставляя три элемента вектора вращения устройства.
TYPE_TEMPERATURE	Измеряет температуру устройства в градусах Цельсия (°C). Этот датчик был заменен датчиком TYPE_AMBIENT_TEMPERATURE начиная с уровня API 14.

Немногие Android-устройства оснащены всеми типами датчиков. Например, большинство смартфонов и планшетов имеют акселерометр и магнитометр, но гораздо меньше устройств имеют барометры или термометры. Кроме того, устройство может иметь более одного датчика данного типа. Например, оно может иметь два датчика силы тяжести, каждый из которых имеет разный диапазон.

Хотя доступность датчика зависит от устройства, она также может варьироваться в зависимости от версии Android. В таблице 3 показана доступность каждого датчика для различных платформ. Указаны только четыре платформы, так как именно на них произошли изменения. Датчики, которые указаны как устаревшие, по-прежнему доступны на последующих платформах (при условии, что датчик присутствует на устройстве).

Таблица 3. Доступность датчика по платформе.

Датчик	Android 4.0 (API 14)	Android 2.3 (API 9)	Android 2.2 (API 8)	Android 1.5 (API 3)
TYPE_ACCELEROMETER	+	+	+	+
TYPE_AMBIENT_ TEMPERATURE	+	—	—	—
TYPE_GRAVITY	+	+	—	—
TYPE_GYROSCOPE	+	+	—	—
TYPE_LIGHT	+	+	+	+
TYPE_LINEAR_ ACCELERATION	+	+	—	—
TYPE_MAGNETIC_FIELD	+	+	+	+
TYPE_ORIENTATION	устарел	устарел	устарел	+
TYPE_PRESSURE	+	+	—	—
TYPE_PROXIMITY	+	+	+	+
TYPE_RELATIVE_ HUMIDITY	+	—	—	—
TYPE_ROTATION_ VECTOR	+	+	—	—
TYPE_TEMPERATURE	устарел	+	+	+

Доступ к этим датчикам и необработанные данные датчиков можно получить с помощью фреймворка сенсоров Android, который является частью пакета android.hardware и включает следующие классы и интерфейсы:

Класс SensorManager используется для создания экземпляра службы датчиков. Этот класс предоставляет различные методы для доступа к датчикам, для регистрации и удаления слушателей событий датчиков. Также предоставляется несколько констант, которые используются для определения точности датчика, установки скорости сбора данных и калибровки датчиков.

Класс Sensor используется для создания экземпляра определенного датчика. Этот класс предоставляет различные методы, позволяющие определять возможности датчика.

Класс SensorEvent используется для создания объекта события датчика, который предоставляет информацию о событии датчика. Объект

события датчика включает в себя следующую информацию: необработанные данные датчика, тип датчика, который сгенерировал событие, точность данных и метку времени для события.

Интерфейс SensorEventListener используется для создания двух методов обратного вызова, которые получают уведомления (события датчика) при изменении значений датчика или при изменении точности датчика.

Обнаружение датчиков и их характеристик

Платформа Android предоставляет несколько методов, которые упрощают определение, какие датчики имеются на устройстве. Также предоставляются методы, позволяющие определять характеристики каждого датчика, такие как его максимальный диапазон, разрешение и требования к мощности.

Чтобы идентифицировать датчики, которые есть на устройстве, необходимо сначала получить ссылку на службу датчиков. Для этого создаётся экземпляр класса SensorManager, вызовом метода getSystemService (), и передаётся аргумент SENSOR_SERVICE. Затем можно получить список всех датчиков на устройстве, вызвав метод getSensorList () и используя константу TYPE_ALL (листинг 1).

Листинг 1. Обнаружение датчиков.

```
public class MainActivity extends ListActivity {
    private SensorManager sensorManager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        sensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
        List<Sensor> deviceSensors =
sensorManager.getSensorList(Sensor.TYPE_ALL);

        List<String> deviceSensorTypes = new ArrayList<>();
        for (int i = 0; i < deviceSensors.size(); i++) {
            deviceSensorTypes.add("Name:
"+deviceSensors.get(i).getName()+"\nType:
"+deviceSensors.get(i).getStringType());
        }
```

```
        setListAdapter(new ArrayAdapter<>(this,
                android.R.layout.simple_list_item_1, deviceSensorTypes));
        getListView().setTextFilterEnabled(true);
    }
}
```

Помимо перечисления датчиков, установленных на устройстве, можно использовать открытые методы класса Sensor для определения характеристик отдельных датчиков. В листинге 1 метод getName () используется для получения имени датчика, а метод getStringType (), доступный начиная с уровня API 20, возвращает тип датчика как строку. Метод getType () возвращает тип датчика как целочисленное значение.

Также можно использовать методы getResolution () и getMaximumRange () для получения разрешения сенсора и максимального диапазона измерения. Вы также можете использовать метод getPower () для получения требований к питанию датчика.

Два открытых метода getVendor () и getVersion () особенно полезны, если нужно оптимизировать приложение для датчиков различных производителей или различных версий датчика.

На рисунке 1 представлен результат работы активити из листинга 1 на устройстве Nokia 2.3.

Помимо основных датчиков, перечисленных в таблице 1, с течением времени в Android добавляется поддержка новых датчиков. Так на рисунке 1, помимо описанных ранее датчиков, также есть новые датчики:

- TYPE_SIGNIFICANT_MOTION (доступен с API 18) — датчик обнаружения значимого движения для пробуждения устройства.
- TYPE_STEP_DETECTOR (доступен с API 19) — датчик шагов. Датчик этого типа запускает событие каждый раз, когда пользователь делает шаг.
- TYPE_STEP_COUNTER (доступен с API 19) — датчик шагов. Датчик этого типа при активации возвращает количество шагов, сделанных пользователем с момента последней перезагрузки.

- TYPE_TILT_DETECTOR — датчик наклона. Датчик этого типа генерирует событие каждый раз, когда обнаруживается событие наклона.
- TYPE_WAKE_GESTURE — датчик жестов пробуждения.
- TYPE_PICK_UP_GESTURE — датчик поднятия. Датчик этого типа срабатывает при поднятии устройства, независимо от того, где оно было раньше (стол, карман, сумка).
- TYPE_STATIONERY_DETECT (доступен с API 24) — датчик, который срабатывает, когда устройство находилось в стационарном состоянии не менее 5 секунд с максимальной задержкой 5 секунд.

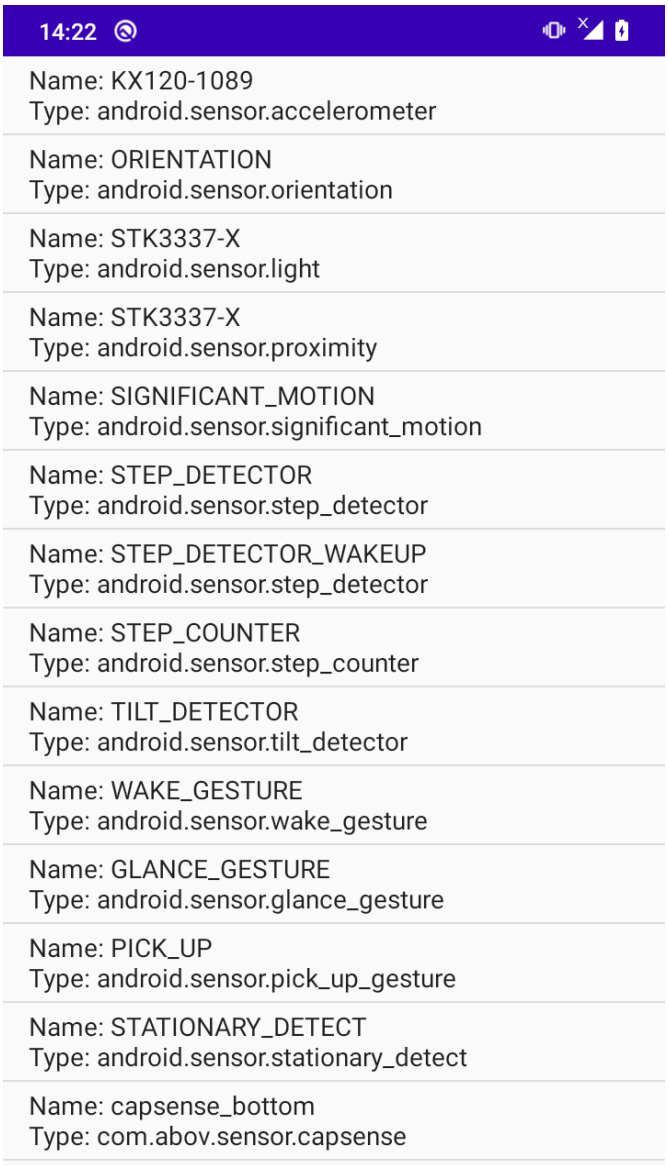

Рисунок 1. Обнаружение всех датчиков на устройстве.

Если требуется определить все датчики конкретного типа, можно передавать в метод getSensorList () определённую константу, например, TYPE_GYROSCOPE или TYPE_GRAVITY.

Также можно определить, существует ли на устройстве датчик определенного типа, используя метод getDefaultSensor () и передав константу типа для определенного датчика. Если устройство имеет более одного датчика данного типа, один из датчиков должен быть назначен датчиком по умолчанию. Если датчик по умолчанию не существует для данного типа, вызов метода возвращает значение null, что означает, что устройство не имеет датчика этого типа. В листинге 2 приведён код, проверяющий, есть ли на устройстве датчики TYPE_GYROSCOPE и TYPE_ACCELEROMETER.

Листинг 2. Обнаружение определённого датчика.

```
public class MainActivity extends AppCompatActivity {
    private SensorManager sensorManager;
    TextView myText;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        myText=findViewById(R.id.text);

        sensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);

        if (sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE) !=
null){
            myText.append("На устройстве есть гироскоп\n");
        } else {
             myText.append("На устройстве нет гироскопа\n");
        }
        if (sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) !=
null){
            myText.append("На устройстве есть акселерометр");
        } else {
            myText.append("На устройстве нет акселерометра");
        }
    }
}
```

На рисунке 2 представлен результат работы активити из листинга 2 на устройстве Nokia 2.3.

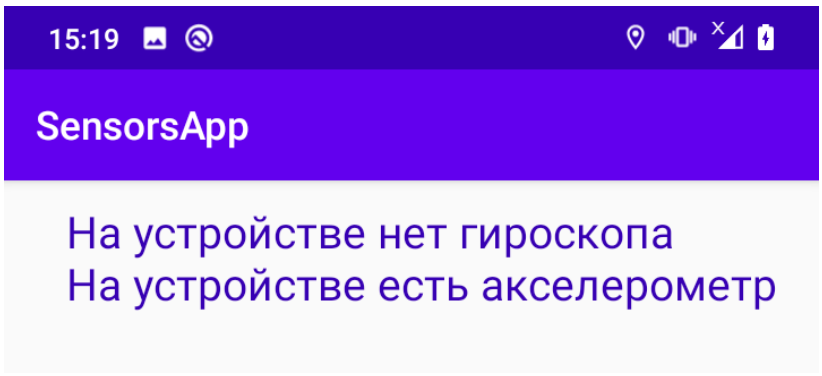

Рисунок 2. Обнаружение определённых датчиков на устройстве.

Мониторинг событий датчика

Для мониторинга необработанных данных датчика необходимо реализовать два метода обратного вызова, которые предоставляются через интерфейс SensorEventListener: onAccuracyChanged () и onSensorChanged ().

Метод onAccuracyChanged () вызывается, если изменяется точность датчика. Метод предоставляет ссылку на измененный объект Sensor и новую точность датчика. Точность представлена одной из четырех констант состояния:

- SENSOR_STATUS_UNRELIABLE ─ значениям, возвращаемым этим датчиком, нельзя доверять, необходима калибровка или окружающая среда не позволяет считывать показания.
- SENSOR_STATUS_ACCURACY_LOW ─ датчик сообщает данные с низкой точностью, необходима калибровка с учетом окружающей среды.
- SENSOR_STATUS_ACCURACY_MEDIUM — датчик сообщает данные со средним уровнем точности, калибровка с учетом окружающей среды может улучшить показания.
- SENSOR_STATUS_ACCURACY_HIGH — датчик передает данные с максимальной точностью.

Метод onSensorChanged () вызывается, если датчик сообщает новое значение. Метод предоставляет объект SensorEvent, который содержит информацию о новых данных датчика, в том числе: точность данных;

датчик, сгенерировавший данные; метку времени, в которую были созданы данные; и новые данные, записанные датчиком.

В листинге 3 показано, как использовать метод onSensorChanged () для отслеживания данных с датчика освещенности. В этом примере необработанные данные датчика отображаются в TextView.

Листинг 3. Мониторинг событий датчика освещенности.

```
public class MainActivity extends AppCompatActivity implements
SensorEventListener {
    private SensorManager sensorManager;
    private Sensor mLight;
    TextView myText;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        myText=findViewById(R.id.text);

        sensorManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
        mLight = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
    }
    @Override
    public final void onAccuracyChanged(Sensor sensor, int accuracy) {
        // Действия, если точность изменилась
    }
    @Override
    public final void onSensorChanged(SensorEvent event) {
        float lux = event.values[0];
        myText.append(Float.toString(lux)+"\n");
    }
    @Override
    protected void onResume() {
        super.onResume();
        sensorManager.registerListener(this, mLight,
SensorManager.SENSOR_DELAY_NORMAL);
    }
    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(this);
    }
}
```

На рисунке 3 представлен результат работы активити из листинга 3 на устройстве. К датчику освещенности подносился фонарик LED Flashlight.

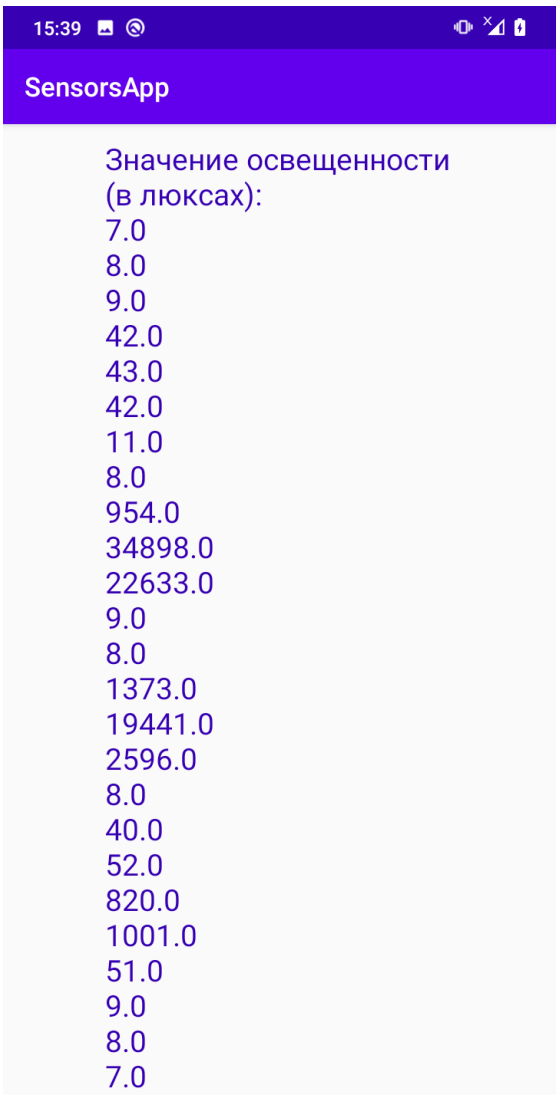

Рисунок 3. Мониторинг событий датчика освещенности.

В данном примере задержка данных по умолчанию (SENSOR_DELAY_NORMAL) указывается при вызове метода registerListener (). Задержка данных (или частота дискретизации) контролирует интервал, с которым события датчика отправляются в приложение с помощью метода обратного вызова onSensorChanged (). Задержка данных по умолчанию подходит для отслеживания типичных изменений ориентации экрана и использует задержку в 200 000 микросекунд (0,2 секунды). Можно указать другие задержки:

- SENSOR_DELAY_GAME (задержка в 20 000 микросекунд),
- SENSOR_DELAY_UI (задержка в 60 000 микросекунд),
- SENSOR_DELAY_FASTEST (задержка в 0 микросекунд).

Начиная с Android 3.0 (уровень API 11), также можно указать задержку как абсолютное значение (в микросекундах).

Указанная задержка является только предполагаемой. Система Android и другие приложения могут изменить эту задержку. Рекомендуется указать наибольшую возможную задержку, поскольку система обычно использует меньшую задержку, чем указанная вами (т.е. следует выбрать самую низкую частоту дискретизации, которая по-прежнему соответствует потребностям приложения). Использование большей задержки снижает нагрузку на процессор и, следовательно, потребляет меньше энергии.

Не существует открытого метода для определения скорости, с которой инфраструктура датчиков отправляет события датчика в приложение, однако можно использовать временные метки, связанные с каждым событием датчика, для расчета частоты дискретизации для нескольких событий.

Данные датчика могут изменяться с высокой скоростью, что означает, что система может вызывать метод onSensorChanged () довольно часто. Лучше всего осуществлять как можно меньше действий в этом методе, чтобы не блокировать его. Если приложение требует, чтобы выполнялась фильтрация или сокращение данных сенсора, это должно выполняться вне метода onSensorChanged ().

Также важно отметить, что в этом примере используются методы обратного вызова onResume () и onPause () для регистрации и отмены регистрации (метод unregisterListener ()) слушателя событий датчика. Рекомендуется всегда отключать датчики, которые вам не нужны, особенно когда активити приостановлено. В противном случае батарея устройства может разрядиться очень быстро, потому что некоторые датчики имеют значительные требования к питанию и могут быстро расходовать заряд батареи. Система не будет автоматически отключать датчики при выключении экрана.

Система координат сенсора

Как правило, при работе с сенсорами используется стандартная трёх-осевая система координат. Для большинства датчиков система координат определяется относительно экрана устройства, когда устройство удерживается в ориентации по умолчанию (рис. 4): ось X горизонтальна и направлена вправо, ось Y вертикальна и направлена вверх, а ось Z направлена к пользователю от экрана. В этой системе координаты за экраном имеют отрицательные значения Z.

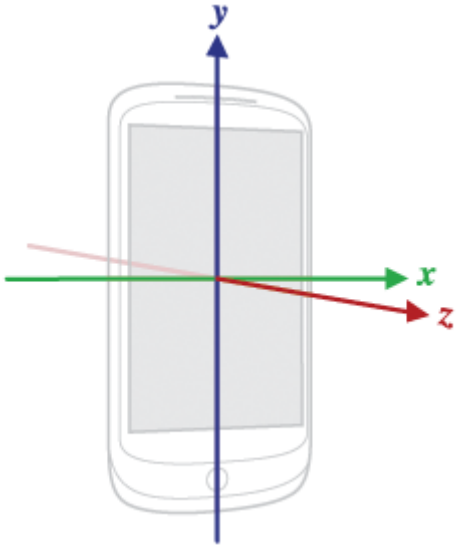

Рисунок 4. Система координат сенсора

Эта система координат используется следующими датчиками:

- Акселерометр (датчик ускорения).
- Датчик силы тяжести.
- Гироскоп.
- Датчик линейного ускорения.
- Датчик геомагнитного поля.

Самым важным моментом, который нужно понять об этой системе координат, является то, что оси не меняются местами при изменении ориентации экрана устройства, то есть система координат датчика никогда не изменяется при перемещении устройства.

Также приложение не должно предполагать, что естественная (по умолчанию) ориентация устройства ― портретная. Естественная ориентация для многих планшетных устройств ― альбомная. А система координат датчика всегда основана на естественной ориентации устройства.

Наконец, если приложение сопоставляет данные датчика с отображением на экране, необходимо использовать метод getRotation () для определения поворота экрана, а затем использовать метод remapCoordinateSystem () для сопоставления координат датчика с координатами экрана. Это требуется, даже если в манифесте приложения указано отображение только портретной ориентации:

```
<activity android:screenOrientation="portrait"
  android:configChanges="orientation|keyboardHidden"></activity>
```

В некоторых датчиках и методах используется система координат, связанная с мировой системой координат (в отличие от системы координат устройства). Эти датчики и методы возвращают данные, которые представляют движение устройства или положение устройства относительно земли.

Датчики движения

Платформа Android предоставляет несколько датчиков, которые позволяют отслеживать движение устройства.

Возможные архитектуры датчиков зависят от типа датчика: датчики силы тяжести, линейного ускорения, вектора вращения, значимого движения, счетчик шагов являются аппаратными или программными.

Датчики акселерометра и гироскопа всегда аппаратные.

Большинство устройств на базе Android имеют акселерометр, а многие теперь оснащены и гироскопом. Доступность программных датчиков более изменчива, поскольку они часто полагаются на один или несколько аппаратных датчиков для получения своих данных. В зависимости от

устройства эти программные датчики могут получать свои данные либо от акселерометра и магнитометра, либо от гироскопа.

Датчики движения полезны для отслеживания движения устройства, например, наклона, встряхивания, вращения или поворота. Движение обычно является отражением прямого ввода пользователя (например, пользователь, управляющий автомобилем в игре, или пользователь, управляющий мячом в игре), но оно также может быть отражением физической среды, в которой находится устройство, например, двигается с вами, пока вы ведете машину.

В первом случае вы отслеживаете движение относительно системы отсчета устройства или системы отсчета приложения; во втором случае вы отслеживаете движение относительно мировой системы координат. Сами по себе датчики движения обычно не используются для отслеживания положения устройства, но их можно использовать с другими датчиками, такими как датчик геомагнитного поля, для определения положения устройства относительно мировой системы координат.

Все датчики движения возвращают многомерные массивы значений датчиков для каждого SensorEvent. Например, во время одного события датчика акселерометр возвращает данные силы ускорения для трех осей координат, а гироскоп возвращает данные скорости вращения для трех осей координат. Эти значения данных возвращаются в виде массива значений с плавающей точкой вместе с другими параметрами SensorEvent.

Датчик вращения и датчик силы тяжести — наиболее часто используемые датчики для обнаружения и мониторинга движения. Датчик вращения универсален и может использоваться для широкого круга задач, связанных с движением, таких как обнаружение жестов, отслеживание изменения угла и отслеживание изменений относительной ориентации. Например, датчик вращения идеально подходит, если вы разрабатываете игру, приложение дополненной реальности, двумерный или трехмерный

компас, или приложение для стабилизации камеры. В большинстве случаев использование этих датчиков является лучшим выбором, чем использование акселерометра и датчика геомагнитного поля или датчика ориентации.

Рассмотрим пример активити, получающего данные с датчика ускорения (акселерометра) и отображающего их в текстовом поле (листинг 4). Датчик ускорения предоставляет трехмерный вектор, представляющий ускорение вдоль каждой оси устройства, включая силу тяжести.

Листинг 4. Мониторинг событий акселерометра.

```
public class MainActivity extends AppCompatActivity implements
SensorEventListener {
    TextView myText;
    SensorManager sensorManager;
    Sensor sensorAccel;
    float[] valuesAccel = new float[3];
    Timer timer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        myText =  findViewById(R.id.myText);
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        sensorAccel =
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }

    @Override
    protected void onResume() {
        super.onResume();
        sensorManager.registerListener(this, sensorAccel,
SensorManager.SENSOR_DELAY_NORMAL);

        timer = new Timer();
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        myText.setText("");
                        myText.append("Ускорение\nX: " + valuesAccel[0] +
"\nY: " + valuesAccel[1] + "\nZ: " + valuesAccel[2]);
```

```
                    }
                });
            }
        };
        timer.schedule(task, 0, 1000);
    }
    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(this);
        timer.cancel();
    }
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {        }
    @Override
    public void onSensorChanged(SensorEvent event) {
         for (int i = 0; i < 3; i++){
             valuesAccel[i] = event.values[i];
         }
    }
}
```

В методе onResume () регистрируем слушатель сенсора. И запускаем таймер, который будет каждые 1000 мсек отображать данные в TextView. В onPause () отписываем слушателя от сенсора, вызывая метод unregisterListener () и отключаем таймер.

На рисунке 5 представлен результат работы активити из листинга 4 на устройстве. Когда устройство лежит на столе, видно, что третья ось (Z), которая в лежачем положении проходит вертикально вверх, показывает ускорение примерно равное гравитации.

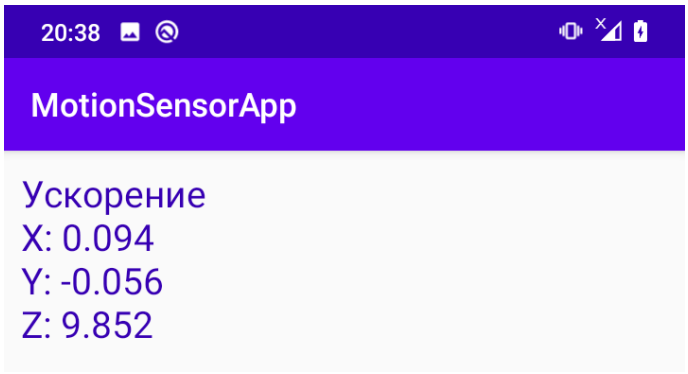

Рисунок 5. Мониторинг событий акселерометра.

Задание

Разработайте приложение, позволяющее определить имеющиеся на устройстве датчики, а также выводящее на экран показания нескольких датчиков.

Контрольные вопросы:

1. Какие три типа сенсоров поддерживает Android?
2. Какие датчики относятся к датчикам движения?
3. Какие датчики относятся к датчикам окружающей среды?
4. Какие датчики относятся к датчикам положения?
5. Чем отличаются программные датчики от аппаратных?
6. Какие распространённые датчики поддерживаются Android? Какие показания они возвращают?
7. Для чего используется класс SensorManager?
8. Для чего используется класс Sensor?
9. Для чего используется класс SensorEvent?
10. Для чего используется интерфейс SensorEventListener?
11. Для чего используется метод getSensorList?
12. Для чего используется метод registerListener?
13. Для чего используется метод unregisterListener?
14. Какие характеристики датчиков можно определить с помощью методов класса Sensor?
15. Когда вызывается метод onAccuracyChanged?
16. Когда вызывается метод onSensorChanged?
17. Какая система координат используется при работе с датчиками Android?