ЛАБОРАТОРНАЯ РАБОТА. ДАТЧИКИ

Цель. Рассмотреть и изучить основы работы с датчиками в ANDROID.

**Краткие сведения**.

В среде Android датчик (sensor) — это аппаратный компонент, встроенный в устройство, который получает данные из окружающей среды и передает их приложениям. В свою очередь, приложения используют данные датчиков для информирования пользователей об окружающей физической среде, управления ходом игры, построения дополненной реальности, реализации полезных инструментов, функционирующих в реальном мире, и т.п. Датчики работают лишь в одном направлении; они предназначены только для чтения.

Следовательно, взаимодействовать с ними совсем несложно. Для получения данных от датчика настраивается слушатель, после чего данные обрабатываются по мере их поступления. Оборудование GPS подобно датчикам, описанным далее.

Ниже перечислены типы датчиков, которые могут быть встроены в устройства Android:

• датчик освещенности;

• датчик приближения;

• датчик температуры;

• датчик давления;

• гироскоп;

• акселерометр;

• датчик магнитного поля;

• датчик ориентации;

• датчик гравитации (начиная с Android 2.3);

• датчик линейного ускорения (начиная с Android 2.3);

• датчик вектора поворота (начиная с Android 2.3);

• датчик относительной влажности воздуха (начиная с Android 4.0);

• датчик NFC (Near Field Communication — коммуникация ближнего поля) (начиная

с Android 2.3).

В файле AndroidManifest.xml можно указать, какими средствами должно располагать устройство для нормальной работы приложения. Если, например, приложению необходим датчик приближения, укажите это в файле манифеста следующим образом:

**<uses-feature android:name="android.hardware.sensor.proximity" />**

Начнем с программы для измерения температуры. Ее текст приведен ниже.

import android.support.v7.app.AppCompatActivity;

import android.hardware.Sensor;

import android.hardware.SensorEvent;

import android.hardware.SensorEventListener;

import android.hardware.SensorManager;

import android.os.Build;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements SensorEventListener {

private TextView temperaturelabel;

private SensorManager mSensorManager;

private Sensor mTemperature;

private final static String NOT\_SUPPORTED\_MESSAGE = "Sorry, sensor not available

for this device.";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

setContentView(R.layout.activity\_main);

temperaturelabel = (TextView) findViewById(R.id.myTemp);

mSensorManager = (SensorManager)getSystemService(SENSOR\_SERVICE);

if(Build.VERSION.SDK\_INT>=Build.VERSION\_CODES.ICE\_CREAM\_SANDWICH){

mTemperature=

mSensorManager.getDefaultSensor(Sensor.TYPE\_AMBIENT\_TEMPERATURE); // requires API

level 14.

}

if (mTemperature == null) {

temperaturelabel.setText(NOT\_SUPPORTED\_MESSAGE);

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

int id = item.getItemId();

if (id == R.id.action\_settings) {

return true;

}

return super.onOptionsItemSelected(item);

}

@Override

protected void onResume() {

super.onResume();

mSensorManager.registerListener(this, mTemperature,

SensorManager.SENSOR\_DELAY\_NORMAL);

}

@Override

protected void onPause() {

super.onPause();

mSensorManager.unregisterListener(this);

}

@Override

public void onSensorChanged(SensorEvent event) {

float ambient\_temperature = event.values[0];

temperaturelabel.setText("Ambient Temperature:\n " +

String.valueOf(ambient\_temperature) + getResources().getString(R.string.celsius));

}

@Override

public void onAccuracyChanged(Sensor sensor, int accuracy) {

// Do something here if sensor accuracy changes.

}

}

++++++++++++++++++++++++++++++++++++++++++++++++ Слушатель устанавливается в методе onResume() активности, а его регистрация отменятся в методе onPause(). Нам не нужно беспокоиться об уровне освещенности, когда приложение не находится на переднем плане.

Методу registerListener() передается значение, представляющее частоту, с

которой необходимо уведомлять об изменениях значения датчика.

Допускаются следующие значения:

• SENSOR\_DELAY\_NORMAL

• SENSOR\_DELAY\_UI • SENSOR\_DELAY\_GAME

• SENSOR\_DELAY\_FASTEST

Важно выбрать подходящее значение для этого параметра. Выбрав SENSOR\_DELAY\_FASTEST, можно превысить возможности приложения по обработке. С другой стороны, некоторые датчики требуют максимально частого чтения;к ним относится, в частности, датчик вектора поворота.

Поскольку активность реализует интерфейс SensorEventListener, в нашем

распоряжении есть два метода обратного вызова для событий датчика:

onAccuracyChanged() и onSensorChanged(). Первый метод вызывается при

изменении точности датчика (или датчиков, т.к. он может быть вызван для

более чем одного датчика). Значением параметра точности (accuracy) может

быть 0, 1, 2 или 3 — соответственно, для ненадежной, низкой, средней или высокой точности. Ненадежная точность не означает, что устройство вышло из строя; обычно она означает, что датчик должен быть откалиброван.

**Акселерометры**

Используя эти датчики, приложение может определить, как устройство физически сориентировано в пространстве относительно направления свободного падения, а также быть в курсе сил, воздействующих на устройство. Предоставление этой информации позволяет приложению выполнять множество интересных функций, от игр до построения дополненной реальности. И, естественно, акселерометры сообщают Android, когда переключать ориентацию пользовательского интерфейса с книжной на альбомную и обратно.

Координатная система акселерометра организована следующим образом. Ось X начинается в нижнем левом углу устройства и направлена вдоль нижней стороны устройства вправо. Ось Y также начинается в нижнем левом углу устройства, но направлена вдоль левой стороны вверх. Ось Z начинается в нижнем левом углу и направлена в пространство вверх от устройства Эта координатная система отличается от используемой при компоновке и в двумерной графике, где начало координат (0, 0) находится в левом верхнем углу, а положительное направление оси Y — вниз по экрану. Чтобы не запутаться при работе с этими координатными системами, следует соблюдать осторожность.

А что собой представляют значения, выдаваемые акселерометром? Ускорение измеряется в метрах за секунду в квадрате (м/с2). Ускорение свободного падения составляет примерно 9,81 м/с2 в направлении центра Земли. С точки зрения акселерометра, значение ускорения свободного падения равно –9,81. Если устройство полностью неподвижно (не перемещается) и находится на идеально гладкой поверхности, показания по осям X и Y будут равны 0, а по оси Z — выглядеть как +9,81. На самом деле значения не будут в точности такими из-за чувствительности и точности акселерометра, но они будут близки к указанным. Когда устройство находится в состоянии покоя, на него воздействует только сила тяжести, а поскольку она направлена строго вниз, и поверхность, на которой лежит устройство, идеально плоская, показания по осям X и Y равны 0. По оси Z акселерометр измеряет силу, воздействующую на устройство, минус сила тяжести. Таким образом, 0 минус –9,81 в результате дает +9.81 — это и будет значением по оси Z (values[2] в объекте SensorEvent).

Значения, посылаемые приложению акселерометром, всегда представляют сумму всех сил, действующих на устройство, минус сила тяжести. Если начать поднимать устройство, первым делом будет изменяться значение Z, т.к. увеличивается сила в направлении вверх (Z). Как только подъем останавливается, общая сила возвращается к одной лишь силе тяжести. Если уронить устройство (теоретически), оно летело бы к земле с ускорением свободного падения, так что акселерометр показал бы в итоге нулевую суммарную силу.

Измерение силы тяжести с помощью акселерометров (Пример кода)

// Файл MainActivity.java

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.widget.TextView;

public class MainActivity extends Activity implements SensorEventListener {

private SensorManager mgr;

private Sensor accelerometer;

private TextView text;

private float[] gravity = new float[3];

private float[] motion = new float[3];

private double ratio;

private double mAngle;

private int counter = 0;

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

mgr = (SensorManager) this.getSystemService(SENSOR\_SERVICE);

accelerometer = mgr.getDefaultSensor(Sensor.TYPE\_ACCELEROMETER);

text = (TextView) findViewById(R.id.text);

}

@Override

protected void onResume() {

mgr.registerListener(this, accelerometer,

SensorManager.SENSOR\_DELAY\_UI);

super.onResume();

}

@Override

protected void onPause() {

mgr.unregisterListener(this, accelerometer);

super.onPause();

}

public void onAccuracyChanged(Sensor sensor, int accuracy) {

// Игнорировать.

}

public void onSensorChanged(SensorEvent event) {

// Использовать фильтр нижних частот для получения силы тяжести.

// motion — это то, что осталось.

for(int i=0; i<3; i++) {

gravity [i] = (float) (0.1 \* event.values[i] +

0.9 \* gravity[i]);

motion[i] = event.values[i] - gravity[i];

}

//ratio — это результат сравнения силы тяжести по оси Y и полной силы тяжести.

// Должно быть не больше 1 и не меньше -1.

ratio = gravity[1]/SensorManager.GRAVITY\_EARTH;

if(ratio > 1.0) ratio = 1.0;

if(ratio < -1.0) ratio = -1.0;

// Преобразовать радианы в градусы, сделать отрицательными

// при направлении вверх.

mAngle = Math.toDegrees(Math.acos(ratio));

if(gravity[2] < 0) {

mAngle = -mAngle;

}

// Отображать каждое 10-е значение.

if(counter++ % 10 == 0) {

String msg = String.format(

"Raw values\nX: %8.4f\nY: %8.4f\nZ: %8.4f\n" +

/\* исходные значения \*/

"Gravity\nX: %8.4f\nY: %8.4f\nZ: %8.4f\n" +

/\* сила тяжести \*/

"Motion\nX: %8.4f\nY: %8.4f\nZ: %8.4f\nAngle: %8.1f",

/\* движение \*/

event.values[0], event.values[1], event.values[2],

gravity[0], gravity[1], gravity[2],

motion[0], motion[1], motion[2],

mAngle);

text.setText(msg);

text.invalidate();

counter=1;

}

}

}

Мы получаем силу тяжести, используя только небольшую часть нового значения из массива event, но крупную часть из предыдущего значения массива gravity. Две используемых части должны давать в сумме 1.0. Мы применяем 0.9 и 0.1. Можете также попробовать другие значения, такие как 0.8 и 0.2. Наш массив gravity не может изменяться так же быстро, как действительные значения датчика. Именно это и делает фильтр нижних частот. Значения массива event будут изменяться, если силы, воздействующие на устройство, приводят к его перемещению, и мы не хотим их измерять как часть силы тяжести.

В массив gravity должны записываться только показатели силы тяжести. Применяемая здесь математика не содержит никакой “магии”, а просто позволяет вычислять значения силы тяжести, которые намного ближе к реальным, чем исходные значения в массиве event.

Обратите внимание в коде на массив motion. Отслеживая отличия между значениями из массива event и вычисленными значениями силы тяжести, в массиве motion мы, в сущности, измеряем активные силы, отличные от силы тяжести.

**ЗАДАНИЕ**.

Напишите приложение с датчиком ACCELERATOR. При перемещении телефона по вертикали вниз (вверх) получайте значение ускорения. Используйте значение ускорения для генерации случайных чисел (придумайте свой способ). Сгенерированные случайные числа выведите в элемент TextView. Таким образом, Ваш ACCELERATOR будет использоваться как генератор случайных чисел!