Assignment 1

Start:	29 September 2014
End:	8 October 2014

Objectives

The goal of this assignment is to familiarize yourself with the Android development process, to think about user interface design, and to learn how to access sensors and actuators on a smartphone. Your application must target Android 4.3 (API level 18). As the emulator cannot emulate all sensors, you will have to test your code on a physical device. Besides implementing the application, you have to answer a mini-test given below. More details are available in the *Deliverables* section.

With this assignment you can gain 10 points out of the total 45.

Task	Points
1	2
2	4
3	2
4	2
Total	10

1 Sensing with Android (2 Points)

Every Android application must provide an Activity as an interface for user interaction. In this exercise, you will let the user access sensors and actuators through a simple user interface.

1. Download and install the Android development toolchain. Now, you can download Eclipse bundled with Android SDK from the following link: `http://developer.android.com/sdk/index.html`. Follow the guidelines of the introduction slides and/or the Android website `http://developer.android.com/sdk/installing/index.html`

2. Create a new Android project called `VS_`*`nethz`*`_Sensors` and select the API 18 as build target. Set the application name to `VS_`*`nethz`*`_Sensors` and the package name to `ch.ethz.inf.vs.a1.`*`nethz`*`.sensors` (*`nethz`* here and also later means the group leader's nethz ID). Create the first Activity and name it `MainActivity`.

3. In the `MainActivity`, design a user interface to list all available sensors of the smartphone. The sensors should be contained in a ListView which automatically resizes with different input sizes. **Hint:** You can retrieve an array of all the available sensors by calling the `getSensorList(Sensor.TYPE_ALL)` method of a `SensorManager` object.

4. Create a second Activity called `SensorActivity`. When the user highlights a sensor in the ListView, the `SensorActivity` should be started through an Intent. The Intent should carry information about the sensor.

5. In the `SensorActivity`, create another ListView that continuously displays the readings for this particular sensor (and not only details about the sensor).

6. Finally, add a Button below the ListView in `MainActivity`. This Button should start another Activity called `ActuatorsActivity`. Implement `ActuatorsActivity` as seen in the Live Hacking Tutorial on the course website and add capabilities to activate the vibration and to play a sound file. For the vibration actuator, offer the user a SeekBar to control the duration.

2 The Anti-Theft Alarm (4 Points)

In this exercise, you will create an application to secure an Android device against unauthorized usage. When the device is armed, movements should be registered. If the user (thief) keeps moving the phone for a certain amount of time, the phone should raise an alarm (e.g., by playing a sound file or sending a silent notification). You should create an Activity to control the sensitivity of the alarm and the timeout after which an alarm is raised, and a Service to deal with the readings from the accelerometer. **You must use the code skeleton provided on the course website which is described below**.

1. Create a new project called `VS_`*nethz*`_AntiTheft` with an Activity called `MainActivity`. The package name should be `ch.ethz.inf.vs.a1.`*nethz*`.antitheft`.

2. The Activity will also need some means to start and stop the background process running the alarm logic. We suggest you to use a ToggleButton to change the state of the alarm.

3. Create a Service called `AntiTheftServiceImpl`. The service must extend the `AbstractAntiTheftService` class provided to you. This abstract class contains method `startAlarm()` which is called to trigger the alarm. The service should run in the background and post an *ongoing* notification which cannot be cleared by the user. This notification should only disappear when the Service is shut down. The notification is used for resuming of the `AntiTheftMain` activity which monitors the state of the Service. **Hint:** `Notification.FLAG_ONGOING_EVENT` and `Notification.FLAG_NO_CLEAR` may be worth a look. Consider the guidelines of the Android website provided at `http://developer.android.com/guide/components/services.html` for services.

4. Create a class which extends `AbstractMovementDetector` class. It implements `SensorEventListener` and overrides `doAlarmLogic(float[] values)` method. This method contains the sensor logic needed to trigger the alarm. Which sensor you use for this is up to you, but we suggest to use the *accelerometer* or the *orientation* sensor. Your logic should recognize a *deliberate* movement (which we will arbitrarily define as a significant change in sensor readings for a period $\Delta_m \geq 5sec$). Accidental movements, i.e., $\Delta_m < 5sec$, should not cause an alarm.

5. The user should have a certain period of time (Δ_t) during which he/she can still disarm the device. This should be done through a notification in the notification bar. You should enable the user to set Δ_t directly in the Activity. This information could be provided by a SeekBar for example and has to be propagated from the Activity to the Service.

6. When Δ_t has elapsed after a deliberate movement, the phone should ring an alarm (i.e., play a sound file). The user should still be able to disarm the device and stop the alarm using the notification mentioned above.

7. Use `Settings.java` class provided to you to store and retrieve constants in the `SharedPreferences`, where you save all the user's preferred settings. The class contains default values for these settings, which you need to adjust in your implementation.

8. Pay attention to typical Android crashes like on rotating the screen, pushing the back button, etc. At the end, do not forget to unregister the sensor event listener. Failing to do so can drain the battery in just a few hours because some sensors have substantial power requirements and can use up battery power quickly. In contrast with earlier Android versions, the system will not disable sensors automatically when the screen turns off.

3 Sensing using Bluetooth Low Energy (2 Points)

In this task, you will use Android's built-in platform support for Bluetooth Low Energy (BLE) to connect to SHTC1 Smart Gadget to sense humidity and temperature. You should display the current measurements of the humidity and temperature sensor to the user.

SHTC1 Smart Gadget is a simple circuit board with a humidity and temperature sensor (referred to as `RH&T` sensor). It can communicate wirelessly with a Bluetooth capable device, like a smart phone, to provide its sensor measurements. Refer to `www.sensirion.com/SHTC1` for details.

Use Android's guide on BLE for implementing this task:
`https://developer.android.com/guide/topics/connectivity/bluetooth-le.html`

1. Create a new project called `VS_`*`nethz`*`_BleSensirion` with an Activity called `MainActivity`. The package name should be `ch.ethz.inf.vs.a1.`*`nethz`*`.ble`.

2. The Activity should first check if BLE is supported on the device. It should then check if bluetooth is enabled, and display a dialog to the user requesting permission to enable it if it is disabled.

3. Implement a scan for available devices. You should limit the time of the scan. Scanning should stop once the user selects one of the listed devices, or the scan-time expires.

4. When the user selects a listed SHTC1 gadget, your activity should connect to it. Implement a `BluetoothGattCallback` class that monitors the connection state. When the connection state changes to `BluetoothProfile.STATE_CONNECTED`, you should attempt to start service discovery on the device.

5. When service discovery succeeds, your `onServicesDiscovered()` method should then iterate through the discovered services to find the temperature_humidity service of the SHTC1 smart gadget. The unique identifier id (UUID) of the `RH&T` service is `0000AA20-0000-1000-8000-00805f9b34fb`. Once the `RH&T` service is found, you should iterate through the characteristics of the service to find the temperature_humidity characteristic. The UUID of the characteristic is `0000AA21-0000-1000-8000-00805f9b34fb`.

6. Normally, you should be able to read the current value of the characteristic. However, the READ property is not set on the sensor, and the `readCharacteristic()` method would return false! To overcome this problem, we build our own characteristic with the permission as follows:

Listing 1: Code to enable READ property for BLE Characteristic.

```java
// set the READ perimission on the characteristic
BluetoothGattCharacteristic rht = new BluetoothGattCharacteristic(
          UUID_RHT_TEMPERATUREHUMIDITY,
          BluetoothGattCharacteristic.PROPERTY_READ
                | BluetoothGattCharacteristic.PROPERTY_NOTIFY,
          BluetoothGattCharacteristic.PERMISSION_READ);

// add the characteristic to the discovered RH&T service
service.addCharacteristic(rht);
```

where `UUID_RHT_TEMPERATUREHUMIDITY` is the UUID of the temperature_humidity characteristic mentioned above.

7. Add the above lines once you discover the `RH&T` service, and before you call `readCharacteristic()` method. Extract the current temperature and humidity values from the raw HEX string, and display them to the user.

4 Mini-Test (2 Points)

As part of the assignment, you should answer the mini-test given below and submit the answers as a PDF file. The test covers general Android knowledge questions, as well as questions related to how you tackled the assignment.

1. (Sensor Framework)
   A) Write a small code snippet to show how to perform the following tasks:

   a) List available sensors on a device
   b) Retrieve the maximum range of a specific sensor
   c) Register for monitoring `accelerometer` sensor changes, at the maximum available rate of acquiring data

   B) The following code snippet is used to monitor the values of the `ACCELEROMETER` and `PROXIMITY` sensors. The values are then passed to the listener activity `MyActivity`. The code for `MyActivity` class is omitted for brevity. **Lines 19-30** in the `onSensorChanged()` method contain a mistake. Point out the mistake, and explain how it may cause a problem.

Listing 2: Code for question 1.b

```
1  import ch.ethz.inf.vs.a1.MyActivity;
2  import android.hardware.Sensor;
3  import android.hardware.SensorEvent;
4  import android.hardware.SensorEventListener;
5
6  public class SensorValuesDetector implements SensorEventListener {
7          float[] proximityValues;
8          float[] accelerometerValues;
9          protected MyActivity listenerActivity;
10
11         public void register(MyActivity activity) {
12                 listenerActivity = activity;
13         }
14
15         @Override
16         public void onSensorChanged(SensorEvent event) {
17           int sensorType = event.sensor.getType();
18
19           switch (sensorType) {
20             case Sensor.TYPE_ACCELEROMETER:
21                 accelerometerValues = event.values;
22                 // notify the listener activity
23                 listenerActivity.displayAccelerometer(getAccelerometerValues());
24                 break;
25             case Sensor.TYPE_PROXIMITY:
26                 proximityValues = event.values;
27                 // notify the listener activity
28                 listenerActivity.displayProximity(getProximityValue());
29                 break;
30           }
31         }
32
```

```
        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {
                // Do Nothing
        }

        public float getProximityValue() {
                return proximityValues[0];
        }

        public float[] getAccelerometerValues() {
                return accelerometerValues;
        }
}
```

2. (Activity lifecycle) Describe what happens to an activity, and which methods are called in the following consecutive scenarios:

   a) An activity `A` is in the foreground, then the user starts another activity `B`.

   b) Activity `A` is no longer visible.

   c) The user navigates back to activity `A`.

3. (Resources) How can you make your Android application automatically handle different screen sizes and densities? How can you reference the layout elements defined in an Android XML file in your code?

4. What are `Intents`? What are they used for? What is the difference between `Explicit Intents` and `Implicit Intents`?

5. (Service lifecycle) State whether each of the following sentences is true or false:

   a) A servcie started by calling `startService()` can never be stopped before it finishes its job.

   b) Both bound and unbound services can interact with several client processes.

   c) A bound service can only be running as long as there is a component bound to it.

   d) Services which perform lengthy computations are automatically started in a separate thread from the main thread.

6. (AndroidManifest file) Suppose that you are implementing an Android application that consists of one activity `MainActivity` and one service `LocationService`. The service retrieves the user's current fine-grained location at regular intervals using Android's `Location Services` framework. The application sends an sms to a phone number with the current location every hour.

   Below is the AndroidManifest file for such application. The file is missing **three** necessary tags for the application to work. List the needed tags.

Listing 3: AndroidManifest file for question 6

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ch.ethz.inf.vs.android.nethz.antitheft"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
```

```
            android:icon="@drawable/ic_launcher"
            android:label="@string/app_name"
            android:theme="@style/AppTheme" >

            <activity
                android:name="ch.ethz.inf.vs.android.nethz.antitheft.MainActivity"
                android:label="@string/app_name" >
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />

                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>

        </application>
    </manifest>
```

Deliverables

The following deliverables have to be submitted by **9:00am, 8 October 2014**:

1. **code.zip** You should create a zip file containing the Eclipse projects created in this assignment. The projects should have been tested both on the mobile phone and on the emulator. The code must compile on our machines as well, so always use relative paths if you add external libraries to your project. Do not forget to include those libraries in the zip file. Please use UTF-8 encoding for your documents and avoid special characters like umlauts.

2. **answers.pdf** Your answers to the mini-test in **PDF** format.

Submission

The deliverables must be uploaded through:
`https://www.vs.inf.ethz.ch/edu/vs/submissions/`
The group leader can upload the files, and other group members have to verify in the online system that they agree with the submission. Use your nethz accounts to log in. The submission script will not allow you to submit any part of this exercise after the deadline. However, you can re-submit as many times as you like until the deadline.