Assignment 1

Start:	30 September 2013
End:	14 October 2013

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. As Android Virtual Devices (AVDs) cannot emulate all sensors, you will have to test your code on a physical device. Besides implementing the application, you have to write a report describing your work. Its detailed requirements are given in the *Deliverables* section. With this assignment you can gain 10 points out of the total 45. The exercises marked with a ⊙ are necessary to meet the minimum requirements ("save-point").

1 Sensing with Android (2 Points)

Every Android application must provide an Activity that is called by the launcher and provides an interface for user interaction. In this exercise you will let the user access sensors and actuators through a simple user interface.

1. Install the toolchain if not already done. The ADT bundle, which includes Eclipse and the Android SDK, is available at: `http://developer.android.com/sdk/index.html`.

2. Create a new Android project called `vs-nethz-sensors` with the package name `ch.ethz.inf.vs.android.nethz.sensors` and a sensible application name of your choice (replace *nethz* with group leader's nETHZ account name). Use the target SDK matching to your phone up to API 17 (4.2.2). Choose a blank Activity for the `MainActivity`.

3. In the `MainActivity`, design a user interface to list all available sensors of your smartphone. The sensors should be contained in a ListView that 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 the information which sensor was selected.

5. In the `SensorActivity`, create another `ListView` that continuously displays the readings for this particular sensor (i.e., not only static 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 Android Tutorial and add capabilities to activate at least the vibration and to play a sound file. For the vibration actuator, offer the user a SeekBar to control the duration.

2 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). In this exercise, we use a Service to deal with the readings from the accelerometer to detect movement. **You must use the code skeleton provided on the course Web site as explained below**.

1. Create a new project called `vs-nethz-antiTheft` with the package name `ch.ethz.inf.vs.android.nethz.antitheft`. Start again with a blank `MainActivity`.

2. The Activity must provide a GUI to control the sensitivity of the alarm and the timeout after which an alarm is raised. It will also need some means to start and stop the background Service running the alarm logic. We suggest you to use a ToggleButton to change the state of the alarm. **Hint:** Use a `PreferenceActivity` to persist your alarm configuration. In case you need to store state during the runtime, have a look at the methods `onSaveInstanceState(Bundle)` and `onRestoreInstanceState(Bundle)` before you start writing to the SD card manually.

3. Create a Service called `AntiTheftServiceImpl`. The service must implement the `AntiTheftService` interface provided to you. This interface contains one method `startAlarm()` which must be called by the sensor logic to trigger the alarm. The service runs in the background and must post an *ongoing* notification, which cannot be cleared by the user. This notification should only disappear when the Service is shut down. Use it to resume the `MainActivity` 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 Web site provided at `http://developer.android.com/guide/components/services.html`.

4. Create a class called `MovementDetector` that extends the `AbstractMovementDetector` class. It must implement the `SensorEventListener` interface and thus provide the `onSensorChanged(SensorEvent event)` method. This should contain your sensor logic needed to trigger the alarm. Which sensor you use for this is up to you, but we suggest to use the *accelerometer* which could be supported by the proximity sensor. Your logic should recognize a *deliberate* movement (which we will arbitrarily define as a change in the sensor readings over 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 will have 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. 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.

8. Comment your code.

3 Enhancements (2 Points)

1. Use the Android 2D graphics library to visualize the retrieved sensor data. For instance, create a graph that shows the accelerometer values over some seconds or the movement while the Anti-Theft Alarm was armed. **Hint:** Take a look at the *ApiDemos* sample files `DrawPoints.java`, `RoundRects.java`, and `PolyToPoly.java` (they are located in the samples folder of the SDK). The demos can be installed using the procedure defined in `http://developer.android.com/tools/samples/index.html`.

2. A problem with the current Anti-Theft Alarm is that the sound can be easily suppressed on many devices by connecting headphones. It could be replaced by a silent alarm (i.e., a text or e-mail message) together with continuous updates of the GPS coordinates to give the owner an idea of the location of the device. In this final section of the assignment, we would like to see you tackle this problem and come up with a creative solution. Your enhancements should be added to `vs-`*`nethz`*`-antitheft` and be clearly marked in the code.

4 Report (2 Points, ⊙)

As part of the assignments, you are required to produce a short, two-page report. You must follow the report template for the assignment, which is provided on the course Web site. Use a scientific writing style which you will also need later for the bachelor's and master's thesis (i.e., formal without narrative elements, but detailed facts).

1. The abstract is a single paragraph that summarizes the key points of the document. For this report, concisely state (i) which Android device you used, (ii) which tasks you completed and which are working correctly or limited, and (iii) what your specific enhancements are.

2. Use the introduction to inform us about your experience with mobile and Android programming. Also report how you distributed the work in your group.

3. In Sections 2 and 3, write about the design and implementation questions of your applications and motivate any choices you have made during the process. Indicate any problems you may have encountered during the development. Please include screenshots and code snippets to explain particular ideas. We encourage you to highlight bits you are especially proud of or do not like at all.

4. When you provide a solution for the final part of this assignment, we expect you to introduce your enhancements and evaluate their usefulness in Section 4.

5. Finally conclude what you have learned in this assignment and summarize the main challenges or eureka moments you encountered.

Deliverables

The following two deliverables have to be submitted by **09:00am, 14 October 2013**:

- **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 (check the `.classpath` file). All libraries must be put into the `libs` directory within the Android project. Please use UTF-8 encoding for your files and avoid special characters such as umlauts in your code.
- **report.pdf** The report in **PDF** format.

The code for the Anti-Theft Alarm part of the assignment **MUST** follow the code skeleton provided to you with the assignment. Marks will be deducted otherwise.

Submission

Report and code must be uploaded through our submission system:

`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 that.