Assignment 1

Start:	01 October 2012
End:	12 October 2012

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 2.3.3 (API level 10). 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 write a report describing your work. More details are available in the *Deliverables* section. With this assignment you can gain 10 points out of the total 45. The exercises marked with a $\odot$ are necessary to meet the minimum requirements (”save point”).

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. We will use the Android 2.3.3 SDK (API level 10) to support the HTC Desire phones. 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 10 as build target. Set the application name to `VS_`*`nethz`*`_Sensors` and the package name to `ch.ethz.inf.vs.android.`*`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 and add capabilities to activate the vibration and to play a sound file. For the first, offer the user a SeekBar to control the duration.

2 The Anti-Theft Alarm (4 Points, $\odot$)

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.

1. Create a new project called `VS_`*`nethz`*`_AntiTheft` with an Activity called `MainActivity`. The package name should be `ch.ethz.inf.vs.android.`*`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 `AntiTheftService`. It should run in the background and host the alarm logic. Your Service should post an *ongoing* notification which cannot be cleared by the user. This notification should only disappear when the Service is shut down. It is to enable 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.

4. Design and implement the sensor logic needed to trigger the alarm. This should all be done inside the Service. 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 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 by creating for example 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 API 10 Demos `DrawPoints.java`, `RoundRects.java`, and `PolyToPoly.java` (they are located in the samples folder of the SDK).

2. A problem with the current Anti-Theft Alarm is that the sound can be easily suppressed by connecting headphones on many devices. 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 provide 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, $\odot$)

As part of the assignments, you should produce a short report **(1-2 pages)**. Please 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. You can include code snippets to explain particular ideas and we encourage you to highlight bits you are especially proud of (or do not like at all). Feel free to also compare the Android platform to other devices such as Symbian or iOS if you have previously gained experience in those. If you provide a solution for the final part of this assignment, we expect you to introduce your enhancements and evaluate their usefulness. A template for your reports will be given on the website of the course.

Deliverables

The following two deliverables have to be submitted by **09:00am, 12 October 2012**:

- **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 files and avoid special characters like umlauts in your code.

- **report.pdf** The report in **PDF** format.

Submission

Report and code 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 that.