Assignment 1

Start:	30 September 2011
End:	10 October 2011

Objectives

The aim of this assignment is to familiarize yourself with the Android development process. In this exercise you will write an application to secure an Android device against unauthorized usage. When the device is armed, movements should be registered. After a certain amount of time, the phone should react to the movements by raising 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 a Service to deal with the output from the accelerometer. Your program must use the Android 2.2 (API 8) and must run on the mobile phones supplied by the department. Besides implementing the application, you are required to produce a document 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 part of the exercise you will enable the user to access sensors and actuators through a simple user interface.

1. Download and install the Android development toolchain. We will use the Android 2.2 SDK to develop for the HTC Desire phones. Follow the guidelines of the introduction slides and/or the Android website `http://developer.android.com/sdk/installing.html`

2. Create a new Android project called `VS_G**_A1_1` and select the API 8 as build target. Set the application name to `VS_G**_A1_1` and the package name to `ch.ethz.inf.vs.android.g**.a1` The `**` always means your own group number. Create the first Activity and name it `SensorsMain`.

3. In the `SensorsMain` Activity, design a user interface to list all available sensors on the mobile phone. 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 on a `SensorManager` object.

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

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

6. Finally, add a Button below the ListView in `SensorsMain`. This Button should start another Activity called `ActuatorsMain`. Implement `ActuatorsMain` as seen in the LiveHacking Tutorial and add capabilities to play a sound file and to activate the vibration. For the latter, offer the user a SeekBar to control the duration.

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

As described in the introduction, the goal of this assignment is to develop an application that provides some sort of protection against unauthorized usage. An Activity should control settings such as the timeout after which an alarm is raised. Please note that the current version of Android 2.2 on the HTC Desire cannot read sensor values when in *standby* mode. For now, we will ignore this limitation.

1. Create a new project called `VS_G**_A1_2` with an Activity called `AntiTheftMain`. The package name should still be `ch.ethz.inf.vs.android.g**.a1` The `**` stands for your group number.

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 Button to toggle 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. **Hint:** `Notification.FLAG_ONGOING_EVENT` and `Notification.FLAG_NO_CLEAR` may be worth a look. This notification is to enable resuming of the `AntiTheftMain` activity that monitors the state of the Service.

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 ($\Delta_m < 5sec$) 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, 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 on typical Android crashes like rotating the screen, pushing the back button, etc. Write comments in the code!

3 Enhancements (2 Points)

As noted in the previous exercise, the current implementation does not provide any security when the device is in *standby* mode. This is somewhat a Catch-22 since we would need to lock the device to provide adequate security against unauthorized users disabling the alarm. One possible solution is to disable standby while the service is running and to *mock up* a separate login screen. Another problem is that the alarm sound can be easily suppressed by connecting headphones. Both problems could be solved by utilizing the GPS chip on the device. GPS data is collected even in standby and it could also be used to provide the owner an idea of the location of the device by the means of a silent alarm (i.e., a text or email message). Additionally, a BroadcastReceiver could be installed to deal with a headset being plugged in. In this final section of the assignment, we would like to see you tackle one of these problems and come up with a creative solution.

However, you may regard the above as suggestions, only. We are interested in any possible enhancements you may come up with! Since this last part is open-ended, any work done here is especially useful to those seeking the highest grade. The enhancements should be added to `VS_G**_A1_2` 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 don't 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, 10 October 2011**:

- **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.
- **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.