Distributed Systems – Assignment 1

Robin Guldener
ETH ID 11-930-369
robing@student.ethz.ch

Nico Previtali
ETH ID XX-XXX-XXX
two@student.ethz.ch

Lukas Bischofberger
ETH ID 11-915-907
lukasbi@student.ethz.ch

ABSTRACT

We developed two mobile applications for the Android platform from the ground up for the HTC Desire Nr. 25. We completed all of the tasks and our apps worked without crashes on the device.

1. INTRODUCTION

For this assignment we implemented two mobile applications on the Android platform using the Android Developer Tools based on Eclipse[2]. For two out of three team members this was the first time working with Android and subesequently the majority of the time spent on the project was devoted to reading the Android API Guides[1] and Android Reference[3]. Whilst the documentation material is mostly very well written, there are a few corner cases where different documents describe methods in mutual disagreement and the provided GUI editing tools of the Android Developer Tools have not always worked to our satisfaction.

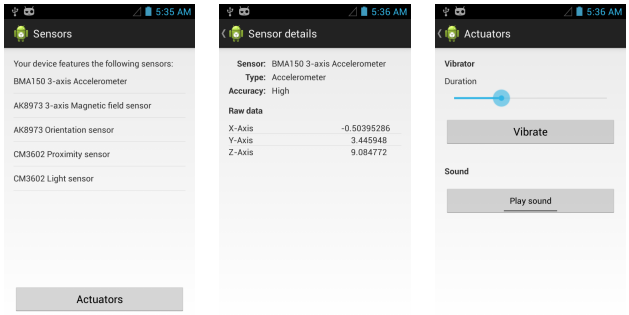
Our team of three was split into two, with Lukas Bischoffberger and Nico Previtali implementing the Anti-Theft Alarm and Robin Guldener implementing the Sensing with Android application. The reporting task was also split accordingly with Robin Guldener additionally covering the Introduction and Conclusion parts whilst Nico Previtali and Lukas Bischoffberger also described their effors for the enhancements of the Anti-Theft Alarm.

2. SENSING WITH ANDROID

The Sensing with Android application (SwA) allows any Android device owner to quickly get an overview of all the sensors available on her device and to read raw data from any sensor in realtime. Additionally, SwA also enables the user to quickly explore the actuators of their device using the builtin Actuators Activity, which supports activating a device's builtin vibrator and playing a predefined, aurally pleasant jingle of bells.

In the following paragraphs each Activity will be presented in detail.

The Main Activity (cf. Figure 1(a)) is the sole entry point of the Application and thus also the only activity that is listed in the launcher. It is the heart of the SwA application and displays a ListView containing all the names of the available sensors. This allows for the efficient selection of any particular sensor. To make the interaction with the list easier for humans with thicker fingers or any potential feline users, we have increased the height of a single row in the list beyond the default value. First tests have shown a particular increase in user-satisfaction, which we could directly link to this design decision. One of the main challenges of the Main Activity is how to pass on the information which sensor was selected to the sensor details activity. We have resolved this issue by using the Java provided hashCode function[4] as detailed in Listing 1. We consider this a particularly elegant solution as it does not depend on any, not necessarily unique, sensor names and leverages existing language design features.



(a) MainActivity (b) SensorsActivity (c) ActuatorsActivity

Figure 1: Activities of the SwA application. Figure (a) shows an example of the sensors available on the lab-provided HTC desire. Figure (b) shows the readings for a particular time of the device's builtin accelerometer. Figure (c) displays the Actuators Activity.

```
1 // s holds a reference to the selected Sensor object
2 Intent intent = new Intent(this, SensorActivity.
      class);
3 intent.putExtra("sensor", s.hashCode());
4 this.startActivity(intent);
```

Listing 1: Passing the selected sensor using Java's hashCode method

The Sensors Activity (cf. Figure 1(b)) provides detailed information on the selected sensor such as the sensor type and provides realtime access to both the raw sensor data as well as the current accuracy of the measured data. To populate the ListView displaying the raw data we have implemented a SensorAdapter, which is a sensor aware implementation of the abstract Adapter class that adjusts the data source exposed to the ListView to the particular sensor currently selected. This provides a clean solution to the heterogenous data we receive from the SensorEvent class[5] and allows for maximum compatability even with future sensor types.

Finally the Actuators Activity (cf. Figure 1(c)) employs a simplistic interface to allow the user to interact with the device's builtin vibrator and play a predefined sound file. The duration of the vibration can be conveniently adjusted using a SeekBar and allows vibration durations ranging from 0ms up to 1000ms, enabling the user to explore different kinds of tactile feedback in a minimal amount of time. Great care was also taken when choosing the builtin sound file and we finally settled on a pleasing, yet very well noticeable jingle of bells.

3. THE ANTI-THEFT ALARM

1. Explain in details the sensor logic you designed which is needed to trigger the alarm. You can also include code snippet as shown in Listing 1.

2. What are the main methods implemented in this part? How do they interact? You can include a state transition diagram like the one shown in Figure 2.

Hint: Just like figures, code listings can convey concise information about your solution. However, you still need to reference and explain them in the text (cf. Listing 2). Only use a listing for really important parts and omit them if it would just be a random part of your code.

```
@Override
protected void onProgressUpdate(final Integer...
    values) {
  textview.setText(index + " done");
  progress.incrementProgressBy(values[0]);
}
```

Listing 2: Descriptive Caption Text

4. ENHANCEMENTS

1. Explain the design/implementation details to visualize the sensors' readings

2. We didn't use a sound alarm for our application. For the local notice we used the vibration of the device. Furthermore we implemented a possibility to alarm the owner of the device through someone else's phone. That means that if an alarm goes off, the app sends a text message to a specified number. Or if there is no number specified we started to implement the possibility to send an email to the phone owner's email address.

5. CONCLUSION

Finally we would like to summarize the main challenges we encountered when implementing assignment 1 and reflect on our key learnings. Clearly the main challenge was getting familiar with the Android platform and understanding its key underlying principles. Whilst we were quickly able to understand the distinction between processes, activities and services, getting used to the GUI layering and figuring out the connection between Layouts, Widgets and their data providers such as Adapters proved much more time consuming than originally anticipated. Additionally the asynchronous nature of the Anti-Theft alarm meant we also had to understand asynchronous function calls and Threading, concepts which can usually be ignored by beginners. Overall we feel we now have a good understanding of the platform and look forward to deepening our knowledge in future assignments.

6. REFERENCES

[1] Android API Guides. `http://developer.android.com/guide/components/index.html`. Accessed on 13 Oct 2013.

[2] Android Developer Tools. `http://developer.android.com/sdk/index.html`. Accessed on 13 Oct 2013.

[3] Android Reference. `http://developer.android.com/reference/packages.html`. Accessed on 13 Oct 2013.

[4] Java Documentation for Object. `http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html`. Accessed on 13 Oct 2013.

[5] SensorEvent API Reference. `http://developer.android.com/reference/android/hardware/SensorEvent.html`. Accessed on 13 Oct 2013.