Università della Svizzera italiana

Faculty of Informatics

Lecture Mobile Computing, fall term 2016
Android Programming Project – Final project report

Group name:	Yet Another Project Team
Group members:	Jacques Dafflon, Samantha Rosso

Instructions

- Please answer the questions reported in the following two pages using the foreseen text boxes.
- Do not change the format, font size or any other elements of this template.
- Be concise and precise.
- Do not exceed the given limit of maximal number of characters. The given limits are intended including white spaces (e.g., the sentence "*This is a nice sentence*" contains 23 characters).
- "The app" mentioned in the questions refers to the Android-based application that you develop in the context of the Mobile Computing class.
- Once compiled, please upload this document to iCorsi.
- The deadline for sending the document is Monday, November 19, 2016 AOE. If you have questions: please post your question(s) in the iCorsi forum.
- IMPORTANT: While the project description referred to you what you *planned* to implement, this final report must describe what you have *actually* implemented.

1. What is the name of your app?

PARC (Phone As a Remote Control)

2. Which problem does the app solve? (Max. 200 characters)

When sitting at your pc, your phone is left unused, PARC solves this by turning it into an extra input for your pc.

3. How does your app solve this problem? (Max. 650 characters)

The app detect hovers of the hand in proximity of the phone. Different motions will trigger different commands which are sent to the computer using UDP over WiFi. A small Python server on the computer will execute those commands.

4. Why is this problem relevant? (Max. 300 characters)

The phone has interesting sensors which can augment the user's interaction with the computer using a keyboard and mouse. So far, those sensors remain unused on the desk right next to the computer.

5. Do other apps exist to solve this (or a very similar) problem?

Yes ☑ No ☐

6. If you answered *yes* to question 5, list the existing apps that are most related to yours and explain how these solutions differ from your own. If you answered *no* to question 5, explain why do you think nobody else has solved this problem before. (Max. 650 characters)

Chrome Remote Desktop and Unified Remote are exampels of apps which allow a user to control the computer from an Android phone. However those try to either replace the keyboard and mouse for a true remote control or emulate buttons on the screen for the user to press. Both options require the user to stop and pick up the phone to perform an action.
The idea behind PARC is to complement the keyboard and mouse and allow the user to blindly use it without picking the phone or even looking at it.

7. Which of the built-in sensors of your phone does the app make use of? (Max. 200 char.)

Proximity, Touch Screen

8. Which of the built-in actuators of your phone does the app make use of? (Max. 200 char.)

WiFi

9. Does your app store sensor data locally, remotely, or in both ways?

Locally ☑ Remotely ☐ Both ☐

10. Motivate your answer to question 9 (i.e., explain why your app stores data and why it does so only locally/remotely or in both ways). (Max. 650 characters)

The app does not store sensor data per say, as readings from the sensor will trigger actions which are sent to the computer using UDP over WiFi. Settings to know which action to send based on the sensor readings will be stored locally on the phone. The IP address and port of the computer are also stored in the settings.
Since those are specific settings for the app which do not need to be shared, it makes sense to simply store them locally on the phone.

11. Does your app use any type of data visualization?

Yes ☐ No ☑

12. If you answered *yes* to question 11, explain which type of data visualization your app offers and why. If you answered *no* to question 11, explain why data visualization is not necessary for your app. (Max. 650 characters)

As mentioned before, the app does not store sensor data, there is nothing much to visualize. Any form of visualization is done on the computer through the execution of the action. For example, the volum up action will increase the volume and increase the volume bar.

13. Does your app perform any type of data processing on the collected sensor data?

Yes ☑ No ☐

14. If you answered *yes* to question 13 explain which type of data processing your app performs on the collected data and why. If you answered *no* to question 13 explain why data processing is not necessary for your app. (Max. 650 characters)

The app will need to process data from the proximity sensor to determine if the hand swiped over the phone or remained static over the phone. It will also need to process data from the different hand gestures that the user can perform, such as the swipe where the phone must determine if the length and velocity of the swipe is enough to be considered a proper swipe. The direction of the swipe (up, down, left or right) also needs to be computed.

15. How do you evaluate whether your app performs correctly and achieves its goal (i.e., solves the problem described in question 2)? (Max. 650 characters)

By measuring the number of time gestures trigger the correct action on the phone. This is working extremely well, in the short time we testeed it we almost never had a gesture miss interpreted. The presentation is a good example where we switched slides back and forth without any problem.

16. Does your app use any third-party services? (E.g., Google maps, Google App Engine, etc.)

Yes ☐ No ☑

17. If you answered *yes* to question 16, list the services your app makes use of. (Max. 200 char.)

18. Which permissions does your app require to be granted by the user? (Max. 200 characters)

INTERNET (to access the wifi socket)

19. What are the main challenges you had to cope with in order to build your app and how did you manage to overcome them? Please also describe "failed attempts", i.e., ideas and solutions you might have explored but did not bring to the expected result. (Max. 1 page)

One of the biggest challenges we faced was implementing the sensors and gestures. Immediately at the beginning, we came to realise that the Bluetooth sensor did not work as expected with OSX. This led to many problems trying to setup a decent Bluetooth connection between phone and computer.

In the end the idea of implementing Bluetooth was scrapped and we moved on to implement a connection using UDP over WiFi and a custom python server. This worked considerably better so we tweaked the application to allow users to set their IP's and ports for connection. There is however a lack of security and when running the server, anyone with your IP and port could potentially send commands from any device, as well as a lack of a discovery service.

The lack of bluetooth also meant we were unable to automatically lock the computer based on the distance between the phone and the computer as this distance should have been computed using the bluetooth signal strength.

Another problem we had was during the implementation of the Proximity sensor which turned out to be too slow for any complex gestures. In the end we settled on two very simple gestures, Swipe Over and Hold Over. Furthermore, the range is rather limited so the user must place their hand extremely close to the sensor for it to be detected. An alternative to this would have been to use the light sensor. However, after extensive testing it turned out to be just as slow and could not recognise a hand under certain light conditions. However, it did have a longer range and under perfect light conditions would work extremely well.

The rest of the application was rather straightforward and the implementation went as expected.

Teamwork:
Considering the small number of activities present in this application, we both worked on the class files together. The sensors were primarily implemented by Jacques and the interface by Samantha. There is however a lot of crossover and we have written around the same number of lines of code.

The Android application was written jointly by both of us. Specifically, Jacques Dafflon wrote most of the UDP-related code while, the main activity, the preferences, and the about and help dialogs were written by Samantha Rosso. The gesture related code was written by both of us together.

The Python server was written by Jacques Dafflon and the images and graphics were made or adapted by Samantha Rosso.