Grundlagen hardwarenahe Programmierung (GHP)

Prof. Dr. Manzke – FH-Kiel – robert.manzke@fh-kiel.de

Laboraufgabe 1: Android Programmierung

Zeitlicher und organisatorischer Ablauf

Das Labor besteht aus 2 Versuchen (Android und Microcontroller) mit jeweils mehreren Unterteilen. Insgesamt gibt es 6 Termine für jede Gruppe. Laborversuche können in Teams mit maximal 2 Personen gelöst werden. Beide Teammitglieder müssen bei der Abnahme in der Lage sein, zu beschreiben, was beide Teammitglieder gemacht haben. D.h. Mitglied A hat bspw. Sensoren programmiert und sowohl Mitglied A als auch B müssen dieses erklären können.

Abgabe von Versuch 1 ist zum 3. Termin fällig, Versuch 2 zum 6. Termin!

Zu den Terminen müssen die jeweiligen Aufgabenabschnitte implementiert, vorgeführt und vom Laborpersonal abgenommen werden.
Aufgrund der Verschachtelung der Vorlesungen mit den Laborgruppen wird es mit Sicherheit zu der Situation kommen, dass Laboraufgaben Stoff verwenden, welcher in der Vorlesungen noch nicht vollständig behandelt wurde. In diesem Regelfall wird von den Studierenden erwartet, sich das fehlende Wissen aus dem Web, Vorlesungsskript oder einschlägigen Büchern selbst anzueignen. Dieser Aufwand fällt unter die Kategorie Selbststudium, wie in der Modulbeschreibung entsprechend vermerkt ist.

Zielsetzung Versuch 1

Im ersten Versuch werden die Grundlagen der Android Programmierung geübt. Es werden einfache GUIs, Debugging-Techniken, Event-Handling, Dynamische UI-Objekterzeugung, Zugriff auf Resourcen, einfaches Threading und Daten-Persistierung behandelt. Außerdem wird auf die Sensorhardware zugegriffen, um den Übergang zur hardwarenahen Programmierung zu schaffen.
Generelle Voraussetzungen für den ersten Versuch sind: Java, Vererbung, Event Handling, Activities, Views, Shared Preferences, LogCat, Toasts, Sensor Interfaces, Threads, … .

Teil A „Hello World“:

1. Installieren Sie die ADT (auf den Laborrechnern bereits erfolgt). Folgen Sie dazu der Anleitung unter http://developer.android.com/sdk/installing/bundle.html

2. Erstellen Sie ein Android Virtual Device (AVD) (auf den Laborrechnern bereits erfolgt). Dieses wird verwendet, um Ihre Anwendungsumgebung zu simulieren.

3. Erstellen Sie ein vertikales LinearLayout mit dem GUI Editor.

4. Dann soll ein Button in den XML Resourcen definiert werden. Beim Klicken des Buttons soll ein Toast ausgegeben werden. Implementieren Sie dazu das Interface OnClickListener in Ihrer Activity.

5. Jetzt soll ein zweiter Button dynamisch im Code in der Activity Methode onCreate() erzeugt werden. Beim Klicken des Buttons soll ein weiterer Toast ausgegeben werden. Definieren Sie dazu Ihren Event Listener diesmal als anonyme Klasse von Typ OnClickListener.

6. Vermerken Sie beide Button Klicks im LogCat.

7. Fügen Sie Ihrem Layout ein EditText Feld hinzu.

8. Überschreiben Sie die Activity Lebenszyklusmethode onStop() und geben Sie eine LogCat Meldung aus.

9. Schreiben Sie in der onStop() Methode Code, der den Textinhalt des EditText Feldes in einem SharedPreferences Objekt ablegt.

10. Erweitern Sie die onCreate() Methode, indem aus den vorher erstellten SharedPreferences der letzte Textinhalt ausgelesen und dem EditText Feld zugewiesen wird.

11. Fügen Sie dem Layout einen weiteren Button hinzu (Layout Editor, XML Resource). Beim Button Klick soll nun eine neue Methode namens createIntent() aufgerufen werden. Definieren Sie dies über die XML Resource-Datei des Layouts.

12. Erstellen Sie eine neue Activity Klasse (Eclipse Template: *Blank Activity*) mit Namen DisplayMessage.

13. Schreiben Sie in der createIntent() Methode nun Code, der als Intent die neue Activity DisplayMessage startet. Übergeben Sie über den Intent der neuen Activity ein Key-Value Paar und stellen Sie dieses in der neuen Activity über eine dynamisch erzeugte TextView dar.

Milestones (vom Laborpersonal abzunehmen)

Milestone	Beschreibung	Abnahme
A1	ADT Entwicklungsumgebung gestartet.	
A2	AVD definiert.	
A3	Linear Layout und Button in Resource definiert.	
A4	Button mit Klick Handler und Toast (XML Resource), Activity mit implementiertem Interface als Klick Handler	
A5	Button mit Klick Handler und Toast dynamisch erzeugt, Interface als anonyme Klasse.	
A6	Klickvermerke im LogCat.	
A7	onStop() überschrieben, LogCat Ausgabe.	
A8	SharedPreferences in onStop() geschrieben.	
A9	SharedPreferences in onCreate() zurückgelesen.	
A10	Button mit Custom-Click-Handler createIntent().	
A11	Neue Activity Klasse Display Message.	
A12	Intent mit neuer Activity und Ausgabe der Übergabeparameter in dynamisch erzeugter TextView.	

Teil B Sensoren

1. Machen Sie sich mit den Grundlagen der Android Sensoransteuerung vertraut: http://developer.android.com/guide/topics/sensors/index.html

2. Erstellen Sie ein neues Android Projekt, das in der Main-Activity die GUI-Elemente Button, sowie drei Progress Bars enthält. Stellen Sie sicher, dass in AndroidManifest.xml die Benutzung der Sensoren eingestellt ist.

3. Testen Sie, ob das Android-Gerät ein Accelerometer- und Lichtsensor hat und initialisieren Sie diese Sensoren.

4. Implementieren Sie das SensorEventListener Interface in Ihrer Activity und registrieren Sie die Listener für die o.g. Sensoren. Achten Sie dabei darauf, Sensorevents nicht mehr abzufangen, wenn die Activity nicht mehr im Benutzerfokus steht (über die Activity Lebenszyklusmethoden).

5. Stellen Sie nun die Orientierungswerte mithilfe der Progress Bars statt.

6. Implementieren Sie einen EventHandler für den Button: Beim Klick soll der Wert des Lichtsensors in einem AlertDialog ausgegeben werden.

Teil C Einfache Threads

1. Erstellen Sie ein neues Android Projekt, mit zwei Buttons, einer Progress Bar und einer TextView

2. Erstellen Sie einen Klick Handler für Button eins, der innerhalb von >5s die ProgressBar bis zum höchsten Stand durchzählt und den Fortschritt prozentual in der TextView darstellt. Prüfen Sie, ob die Oberfäche während des Zählens noch auf Benutzereingaben reagiert (klicken Sie bspw. Button zwei).

3. Erweitern Sie Ihr Programm um Threads. Das Hochzählen soll nun nach Klick auf den Button in einem eigenen Thread geschehen (Worker Thread). Aktualisieren Sie auch hier die ProgressBar und die TextView entsprechend Aufgabe 2 direkt aus dem neuen Thread. Dokumentieren Sie das Ergebnis.

4. Stellen Sie nun sicher, dass Sie nicht aus dem neuen Thread die ProgressBar und die TextView updaten, sondern hierfür einen Handler schreiben (vgl. Zugriffe auf UI Elemente außerhalb des UI Threads sind verboten!). Prüfen Sie erneut das Reaktionsverhalten der Oberfläche.

5. Implementieren Sie einen Worker Thread mithilfe des „AsyncTask“ Interfaces. Der Worker Thread könnte bspw. Resourcen aus dem Web laden oder eine zeitaufwändige Berechnungen durchführen (bspw. Primzahlen berechnen). Die Ergebnisse des Worker Threads sollen dann dem UI Thread wieder zur Verfügung gestellt werden.

Teil D Optionale Aufgaben

1. Implementieren Sie ein Programm zur Auswertung der Orientierungswerte. Es soll eine geordnete Liste erstellt werden, die eine Zeithistorie der Veränderung der Orientierung darstellt (Änderungsevents müssen generiert werden). Sammeln Sie nur Orientierungsänderungen über einem gewissen Grenzwert (Tipp: ein laufender Mittelwert kann hier hilfreich sein, um Rauschen zu unterdrücken und nur sinnvolle Änderungen zu sammeln). Die Liste würde bspw. folgendermaßen aussehen:
   0s → original orientation
   5s → tilted (10, 90, 0)°
   14s → tilted back to original orientation
   …

2. Erkundigen sie sich über das “android.bluetooth“ Package. Implementieren sie eine einfache BlueTooth Kommunikation unter Verwendung eines BlueTooth Low Energy (BLE) Adapters. Sprechen sie das Laborpersonal zu weiteren Details an.

Milestones (vom Laborpersonal abzunehmen)

Aufgabenteil	Beschreibung	Abnahme
B:2	Android Projekt mit Button und Progress Bars.	
B:2	Sensoren eingestellt im AndroidManifest.xml.	
B:3	Programmatischer Test, ob Sensoren vorhanden.	
B:4	Sensor Interfaces implementiert und Listener registriert.	
B:5	Orientierungswerte in Progress Bars dargestellt.	
B:6	EventHandler für Button und Darstellung der Lichtsensorwerte.	
C:1	Android Projekt mit Button und Progress Bar.	
C:2	Hochzählen ohne Thread.	
C:3	Hochzählen mit Thread.	
C:4	Worker Thread mit Async Task Interface.	