Technische Hochschule Nürnberg

IT-Projekt

Verteiltes Erdbebenwarnsystem

Bearbeitungszeitraum: SS13 - WS13/14

IT Projekt

vorgelegt von	Christopher Althaus
	Baris Akdag
	Niklas Schäfer
	Benjamin Brandt
	Jürgen Hetzel
Betreuer	Prof. Dr. Michael Zapf
Abgabe:	14. Februar 2014

Erklärung

Hiermit versichern wir, dass wir die Arbeit selbständig verfasst, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet zu haben.

_____

Christopher Althaus

_____

Baris Akdag

_____

Niklas Schäfer

_____

Benjamin Brandt

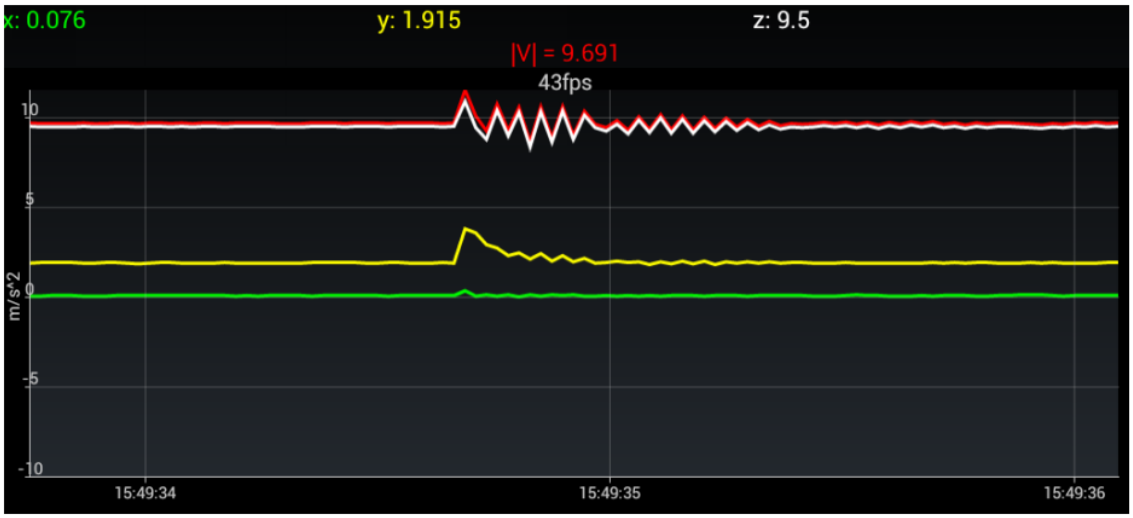
_____

Jürgen Hetzel

Nürnberg, den 25. Januar 2014

Abstract

Portable Geräte wie aktuelle Smartphones und Tablet-Computer besitzen in der Regel eine Vielzahl von Sensoren, darunter auch solche, die Beschleunigungen feststellen können. Diese können insbesondere genutzt werden, um Erschütterungen des Geräts festzustellen. Die unten gezeigte Grafik ist die Ausgabe einer Android-Anwendung (App), welche diese Sensoren ausliest.



Offensichtlich werden diese Sensoren ständig ausgelöst, wenn der Benutzer das Gerät mit sich führt, während er sich fortbewegt. Dabei sind die Werte der Sensoren unmittelbar von der individuellen Bewegung abhängig und daher stets zwischen zwei Geräten verschieden.

Interessant wäre es, wenn es möglich wäre, Korrelationen zwischen den Sensorwerten auf verschiedenen Geräten zu finden. Dies würde darauf hindeuten, dass beide Geräte, zumal wenn sie an verschiedenen Orten aufbewahrt werden, dasselbe Ereignis wahrgenommen haben, etwa eine Erschütterung im Boden.

Dies könnte dazu genutzt werden, um ein automatisches Erdbebenmeldesystem zu realisieren. Wenn eine gewisse Menge von Geräten zur gleichen Zeit ein ähnliches Erschütterungsmuster detektieren, ist davon auszugehen, dass sich ein Erdbeben ereignet. Dies wird natürlich von den Anwendern selbst auch bemerkt werden, jedoch könnten die Geräte einerseits einen Alarm auslösen, der auch solche Menschen warnt, die aus diversen Gründen das Ereignis nicht wahrnehmen (schlafen oder im Auto sitzen), andererseits könnten Sicherheitsmaßnahmen in Gang gesetzt werden (automatisches Abstellen der Gasversorgung, Abstellen des Stroms an gefährlichen Orten usw.).

Inhaltsverzeichnis

1 Teamorganisation

Die Projektgruppe besteht aus Niklas Schäfer, Baris Akdag, Christopher Althaus, Benjamin Brandt sowie Jürgen Hetzel. Innerhalb der Gruppe sind zu Beginn die verschiedenen Aufgabengebiete nach Interessen und Fähigkeiten des einzelnen verteilt worden.
Baris Akdag verfügte im Vorfeld Fachkenntnisse in den Bereichen WebServices und Datenbanken. Er übernahm die Entwicklung des gesamten WebServices inklusive Datenbank und Bereitstellung. Durch die Erfahrung von Christopher Althaus im Bereich Android Programmierung bot er sich neben Niklas Schäfer an, die Android Applikation zu Entwicklen.
Dabei übernahm Niklas Schäfer als Hauptaufgaben die Lokalisierung des Geräts, die Einbindung der Google Maps Karte und die Implementierung von Einstellmöglichkeiten innerhalb der App. Christopher Althaus widmete sich neben der groben Strukturierung der App hauptsächlich um die Benutzerbenachrichtigung im Falle eines Erdbebens und um die Aufgabengebiete rund um den Beschleunigungssensor. Diese umfassen zum einen die Erdbebenerkennung innerhalb der Applikation und zum anderen die Einbindung eines Diagramms zur Visualisierung der Beschleunigungsdaten. Jürgen Hetzel und Benjamin Brand übernahmen während des Projektablaufs einen Großteil der Literaturrecherche. Ebenso kümmerte sich Jürgen Hetzel zum Ende des Projekts um das Refactoring der Android Applikation. Da Benjamin Brand über eine große Auswahl von Geräten verfügte, übernahm er zusätzlich das Testen der Anwendung.
Über den gesamten Zeitraum der Bearbeitung ist eine enge Zusammenarbeit und gute Kommunikation Grundlage für ein erfolgreiches Umsetzen des Projekts gewesen.

2 Motivation

Sucht man im Internet Nachrichten über das Thema Erdbeben, wird schnell ersichtlich, dass beinahe jeden Tag ein ernstzunehmendes Erdbeben auftritt. Sucht man weiterhin nach einer zuverlässigen Vorhersage für Erdbeben, stellt man ebenso schnell fest, dass dies zurzeit noch nicht möglich ist.

Mittels der Umsetzung eines verteilten Erdbebenwarnsystems ist die Warnung zwar auch erst möglich, wenn das Erdbeben bereits spürbar ist, da sich Erdbeben jedoch vom Epizentrum aus ausbreiten, können umliegende Bereiche noch von einer Warnung profitieren. Zudem könnten, wie bereits im Abstract beschrieben, Sicherheitsmaßnahmen in Gang gesetzt werden. Für das verteilte Erdbebensystem ist Android als Plattform ausgewählt worden. Dies begründet sich in der großen Verbreitung des Systems, welche momentan bei über 64% weltweit liegt[1].

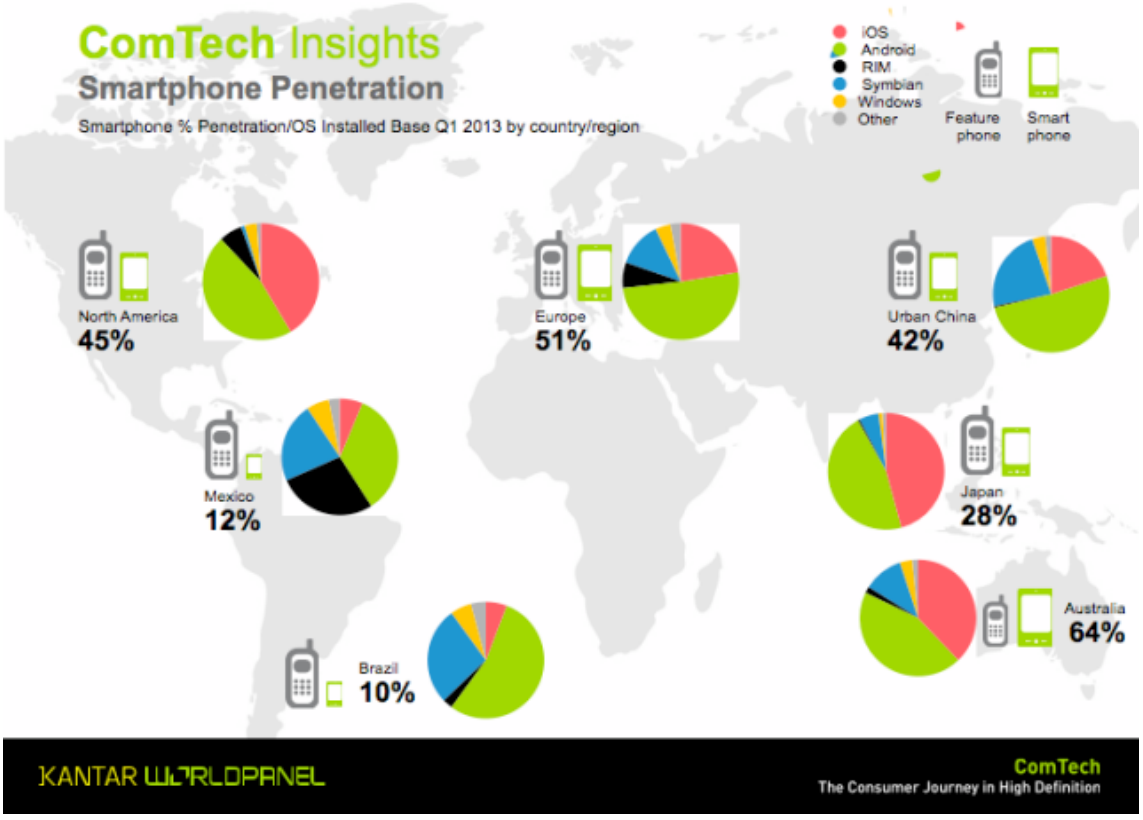


Abbildung 1: Smartphone Betriebssystem-Verbreitung

[1] http://techcrunch.com/2013/04/28/android-picks-up-the-pace-in-smartphone-sales-over-ios-globally-while-windows-phone-continues-with-modest-gains-says-kantar/

3 System Struktur

Die Erdbebenerkennung soll über ein verteiltes System erfolgen. Prinzipiell handelt es sich hierbei um ein Client-Server-System, wobei die Android Smartphones die Clients darstellen. Den Teil des Servers soll ein WebService übernehmen. Die die Strukturierung und Kommunikationsbeziehung dieser beiden Komponenten ist in Abbildung 2 dargestellt.

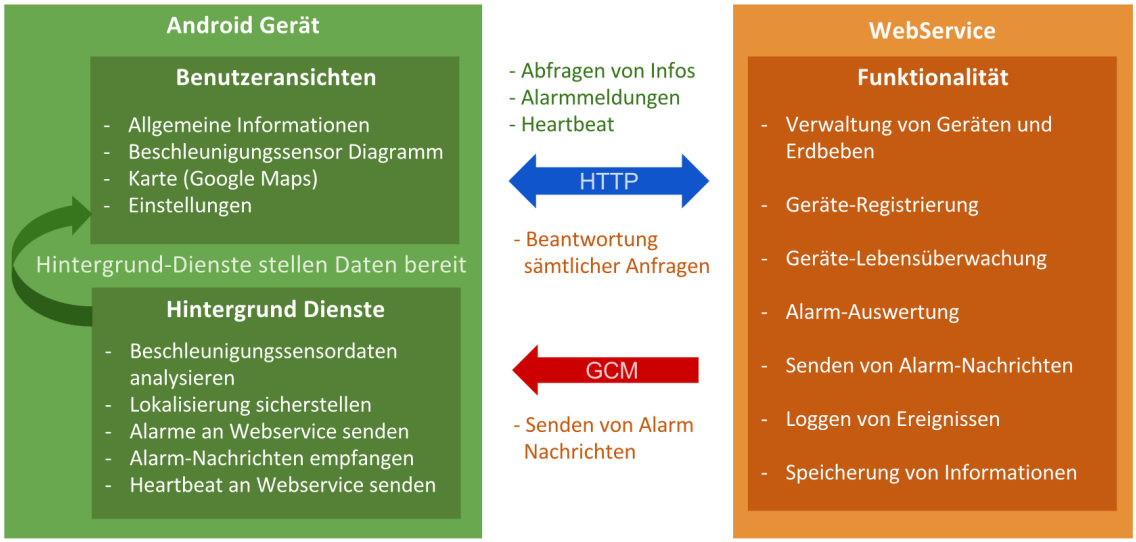


Abbildung 2: Struktur des Projektes

Das rechts dargestellte Android Gerät kann grundlegend in die Benutzeransichten und Hintergrunddienste unterteilt werden.
Die Benutzeransichten stellen dabei den für den Nutzer der App sichtbaren Teil dar. Hierzu gehören zum einen allgemeine Informationen, wie das letzte erkannte Erdbeben oder die Anzahl der verbundenen Geräte. Weiterhin soll dem Nutzer ein Diagramm angezeigt werden, welches die Daten des Beschleunigungssensors anzeigt, um dem Benutzer ersichtlich zu machen, dass die Sensorauswertung funktioniert. Ebenso soll innerhalb der Applikation eine Karte angezeigt werden, welche die verbundenen Geräte und die erkannten Erdbeben für den Nutzer visuell darstellt.
Um die Applikation für den Nutzer anpassbar zu machen, soll ein Einstellungsmenü existieren, in dem beispielsweise die Alarmeinstellungen geändert werden können.
Neben diesen sichtbaren Anteilen bedarf es zahlreicher Hintergrundaktivitäten, welche die eigentlichen Aufgaben der Applikation bewerkstelligen. Zu den wohl Wichtigsten zählen hier die Lokalisierung des Gerätes und die Auswertung der Beschleunigungssensordaten. Diese Dienste sollen beim Systemstart des Smartphones automatisch im Hintergrund gestartet werden.
Weitere Aufgaben der im Hintergrund laufenden Dienste sind das Senden von Alarmen an den WebService, falls die Erdbebenerkennung meint, ein Erdbeben erkannt zu haben und das Empfangen und Verarbeiten von Alarm-Nachrichten des Webservice. Ebenso soll die Applikation in regelmäßigen Abständen eine Art Ping an den WebService schicken, damit dieser weiß, welche

Geräte noch aktiv sind.

Da die Hintergrunddienste Daten verwenden, welche auch in den Benutzeransichten benötigt werden, wie beispielsweise die Beschleunigungssensordaten für das Diagramm, sollen diese Daten für die Benutzeransichten bereitgestellt werden und somit nicht doppelt vom System abgefragt werden.

Der in der Abbildung 2 rechts dargestellte WebService dient somit der Verwaltung aller Geräte. Diese Verwaltung soll neben den Geräten auch die gemeldeten Erdbeben umfassen. Zur Verwaltung dieser Informationen soll eine Datenbank benutzt werden.

Die eigentliche Funktionalität aus Sicht des Android Gerätes, welche der WebService bereitstellt, ist die Geräte-Registrierung, die Geräte-Lebensüberwachung und die Alarmauswertung. Die Geräte-Registrierung soll dabei vom Nutzer unbemerkt beim ersten Aufrufen der App geschehen. Die Geräte-Lebensüberwachung dient, wie bereits beschrieben, dazu, dass der WebService Kenntnis darüber hat, welche Geräte noch aktiv sind. Wichtig wird dies bei der Alarmauswertung. Dabei soll nach einem eingehendem Alarm nach dem Mehrheitsprinzip ausgewertet werden, ob es sich um einen Fehlalarm handelt oder nicht. Da jedoch nur Geräte, welche meinen ein Erdbeben erkannt zu haben, eine Meldung an den WebService schicken, ist es wichtig zu wissen, welche Geräte in der Nähe noch aktiv sind, die keine Meldung geschickt haben.

Um in der Entwicklungsphase Fehler schneller erkennen zu können, soll innerhalb des WebService eingehende Ereignisse, wie beispielsweise eine Alarmnachricht eines Gerätes, geloggt werden. Wie in der Abbildung 2 ersichtlich, können das Android Gerät und der WebService mittels einer HTTP Verbindung bidirektional kommunizieren. Innerhalb der Android Anwendung sollen alle Anfragen, Alarmmeldungen und auch die Lebenszeichenüberwachung mittels HTTP an den Service geschickt werden und daraufhin auch mittels HTTP beantwortet werden.

Wie abgebildet, nutzt der WebService zur Alarmierung der Android-Geräte im Falle eines Erdbebens eine Verbindung namens GCM. GCM steht hierbei für Google Cloud Messaging und erlaubt es, Nachrichten an ein Android Gerät zu verschicken, ohne dabei eine extra Verbindung herstellen zu müssen.

4 RESTful WebService

Todo.

5 Erdbebenerkennung unter Android

Die Erdbebenerkennung innerhalb von Android ist ein wesentlicher Bestandteil des gesamten Systems. Die Realisierung folgt dabei im wesentlichen zwei Grundprinzipien. Zum einen soll die Empfindlichkeit bewusst hoch sein, zum anderen soll der Akkuverbrauch so gering wie möglich gehalten werden.
Die hohe Empfindlichkeit ist darin begründet, dass es besser ist einen Fehlalarm auszulösen, als ein reales Erdbeben nicht als solches zu erkennen. Der möglichst geringe Akkuverbrauch ist der Benutzerakzeptanz der Anwendung zuzuschreiben. Kein Anwender möchte eine Software installieren, welche den Akku in kürzester Zeit leert. Um den Akkuverbrauch gering zu halten, bedarf es somit eines möglichst einfach aufgebautem Erkennungsalgorithmus für Erdbeben.

5.1 Abfragen der Sensordaten unter Android

Innerhalb der Android Anwendung werden die Sensordaten in einem Service im Hintergrund abgefragt. Dieser Service nutzt dazu den sogenannten *SensorManager* aus dem *android.hardware* Paket. Mittels des *SensorManagers* ist es möglich, auf alle Sensoren eines Android Gerätes zuzugreifen. Daraufhin wird ein neuer Sensor vom Typ *Accelerometer* angelegt, welcher daraufhin im *SensorManager* registriert werden kann. Im Listing 1 ist der dafür nötige Code aufgeführt.

Listing 1: Abfragen der Sensordaten unter Android

```
1  public final String ACCEL_SAMPLE = "com.th.nuernberg.quakedetec.ACCEL_SAMPLE";
2  public final String ACCEL_SAMPLE_KEY = "ACCELERATION_SAMPLE";
3  SensorManager sensManager = getSystemService(Context.SENSOR_SERVICE);
4  Sensor accel = sensManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
5  sensManager.registerListener(sensListener, accel, SensorManager.SENSOR_DELAY_UI);
6
7  private final SensorEventListener sensListener = new SensorEventListener() {
8      public void onSensorChanged(SensorEvent e) {
9          float x = e.values[0];
10         float y = e.values[1];
11         float z = e.values[2];
12         long t = System.currentTimeMillis();
13         AccelSample accelSample = new AccelSample(x, y, z, t);
14         broadcastSingleSample(accelSample);
```

```
15     }
16 };
17
18
19 protected void broadcastSingleSample(AccelSample sample) {
20     Intent intent = new Intent(ACCEL_SAMPLE);
21     intent.putExtra(ACCEL_SAMPLE_KEY, (Parcelable) sample);
22     sendBroadcast(intent);
23 }
```

Zunächst wird, wie bereits geschrieben, der *SensorManager* und die Variable accel vom Typ *Sensor* als Beschleunigungssensor initialisiert. Daraufhin wird mittels des *SensorManagers* ein *SensorEventListener* namens *sensListener* für den angelegten Beschleunigungssensor registriert. Dieser *SensorEventListener* wird daraufhin bei jedem Event des Beschleunigungssensor aufgerufen. Beim Registrieren des Sensors wird zudem die Aktualisierungsrate mit angegeben. Diese ist momentan auf den Wert *SENSOR_DELAY_UI* eingestellt, was einer eher mittleren Aktualisierungsrate entspricht. Würde die Rate jedoch höher gesetzt werden, würde dies den Akkuverbrauch in die Höhe treiben. Die nächst niedrigere Stufe wäre *SENSOR_DELAY_NORMAL*. Sie wird innerhalb von Android zur Displayausrichtung genutzt und hat somit eine zu langsame Aktualisierungsrate, dass sie zur Erkennung von Erdbeben nicht geeignet wäre.

Im *SensorEventListener* werden nun die aktuellen Sensorwerte entgegengenommen. Dabei handelt es sich um die Beschleunigungswerte der X-, Y- und Z-Achse. Die Sensordaten werden nun zusammen mit der aktuellen Systemzeit in einer Variable vom Typ *AccelSample* gespeichert. Diese Klasse dient lediglich dazu, die Sensordaten eines Zeitpunktes innerhalb des Systems mittels eines Broadcasts zu verbreiten. Dieser Broadcast wird von der Methode *broadcastSingleSample* ausgelöst.

Prinzipiell könnte auch innerhalb des *SensorEventListeners* die Auswertung der Beschleunigungssensordaten stattfinden. Da jedoch neben der Erdbebenerkennung auch das Diagramm, welches der Nutzer innerhalb der App sieht, diese Daten benötigt, ist es der übliche Weg, einmal innerhalb einer Applikation die Daten vom Sensor anzufragen und daraufhin innerhalb der Anwendung diese Daten mittels eines Broadcast zu verteilen.

Wie in der Implementierung der *broadcastSingleSample* Methode zu sehen, wird innerhalb dieser Methode ein Intent angelegt. Intents werden vom Android Betriebssystem für den asynchronen Austausch von Nachrichten verwendet. Dem angelegten Intent wird dabei der String *ACCEL_SAMPLE* als Argument übergeben. Hierbei handelt es sich um eine Art Schlüssel für die auszuführende Aktion des Intents, welcher später dazu benutzt werden kann, Intents dieses Schlüssels abzufangen und zu verwenden. Ebenso werden dem Intent die Beschleunigungssensordaten angehangen. Dies geschieht mittels einer Key/Value Angabe. Dabei ist der String *ACCEL_SAMPLE_KEY* der Schlüssel für die übergebenen Beschleunigungssensordaten der Variable sample. Mittels des hier angegebe-

nen Schlüssels können Empfänger dieses Intents, die Daten, welche ihm hinzugefügt worden sind, zweifelsfrei zuordnen. Zum Ende der Methode wird das Intent als Broadcast verschickt.
Möchte nun eine Activity oder ein anderer Service auf die Daten des Intents zugreifen, wird ein Broadcast-Receiver benötigt, welcher auf Intents mit dem Schlüssel aus der Variable *ACCEL_SAMPLE* wartet. Eine beispielhafte Implementierung eines solchen Broadcast-Receivers ist dabei im nachfolgenden Listing 2 dargestellt.

Listing 2: Broadcast-Receiver zum Auslesen der Beschleunigungssensordaten

```
1  private class AccelerationBroadcastReceiver extends BroadcastReceiver {
2      @Override
3      public void onReceive(Context context, Intent intent) {
4          if (intent.getAction().equals(Accelerometer.ACCEL_SAMPLE)) {
5              AccelSample sample =
                   intent.getParcelableExtra(Accelerometer.ACCEL_SAMPLE_KEY);
6              if (sample != null) {
7              //Erdbebenauswertung
8              }
9          }
10     }
11 }
```

Wie im Quellcodeauszug ersichtlich, empfängt ein Broadcast-Receiver zunächst jegliche auftretende Intents. Jedoch kann unter der Verwendung des Schlüssels, welcher dem Intent zugewiesen worden ist, jedes eingehende Intent dahingehend überprüft werden, ob es sich um das gefragte Intent handelt oder nicht. Ist dies der Fall, so können die dem Intent angehangenen Daten ausgelesen werden. Hierzu wird der Schlüssel verwendet, welcher dem angehangenen Inhalt zugewiesen worden ist. In diesem Fall also der Wert von *ACCEL_SAMPLE_KEY*, der Klasse *Accelerometer*. Mit dem ausgelesenen Wert des Intents stehen nun im Broadcast-Receiver die Beschleunigungswerte eines einzelnen Zeitpunktes zur Verfügung. Somit können diese für die Erdbebenauswertung genutzt werden, welche im folgenden erläutert werden soll.

5.2 Erdbebenauswertung

Wie bereits erwähnt, soll der zur Auswertung benutzte Algorithmus möglichst ressourcenschonend implementiert werden. Deshalb ist gleich zu Beginn der Implementierung beschlossen worden, dass die Erkennung nicht alle drei Achsen (X, Y, Z) auswerten soll, sondern den Betrag aller Beschleunigungen.
Prinzipiell kann man sich die vom Beschleunigungssensor erhaltenen Werte als Vektoren vorstellen, welche einen rechtwinkligen Raum aufspannen. Möchte man nun den Betragswert dieser Vektoren

bestimmen, genügt es, die Raumdiagonale r dieses Raumes zu berechnen. Diese kann mit der Formel $|r| = \sqrt{x^2 + y^2 + z^2}$ berechnet werden. Die Formel beruht dabei auf dem Satz des Pythagoras. Zunächst wird dabei die Diagonale der Grundfläche, welche durch zwei Achsen aufgespannt wird, berechnet ($d = \sqrt{x^2 + y^2}$). Daraufhin wird der Satz des Pythagoras ein erneutes Mal auf die verbleibende Achse angewendet ($|r| = \sqrt{d^2 + z^2}$). Durch das Quadrieren von d fällt dessen Wurzel weg, womit sich die oben genannte Formel ergibt.

Da auf der Erde stets eine Erdanziehungskraft von etwa 9,81 $\frac{m}{s^2}$ wirkt, liegt der berechnete Betragswert der Beschleunigung bei einem still liegendem Gerät stets bei etwa 9,81 $\frac{m}{s^2}$. Um die Erkennung von Beschleunigungsänderungen zu vereinfachen, ist deshalb entschieden worden, die Erdbeschleunigung vom Betragswert abzuziehen. Auf diese Weise ergibt sich letztendlich die Formel $|a| = \sqrt{x^2 + y^2 + z^2} - 9,81 \frac{m}{s^2}$ zur Berechnung der betragsmäßigen Beschleunigung des Gerätes. Wird nun das Gerät bewegt, ergeben sich Ausschläge um den Nullpunkt herum, wie in Abbildung 3 dargestellt.

Abbildung 3: Beschleunigungssignal nach der Zusammenfassung

Die dargestellten Ausschläge sind eine Aufzeichnung des Beschleunigungssignals, während das Gerät mit erdbebenartigen Schlägen bewegt worden ist. Durch die Rüttelbewegung des Gerätes wirkt darauf abwechselnd eine Beschleunigung in Richtung der Erdanziehung, wodurch ein positiver Ausschlag entsteht und daraufhin eine Beschleunigung entgegengesetzt der Erdanziehung. Dadurch wird die Erdanziehung aufgehoben und es entsteht ein negativer Ausschlag.

Da bei Erdbeben in kurzer Zeit eine hohe Anzahl von Schlägen auf das Gerät einwirken, ist es in Anbetracht dieser Tatsachen möglich, das Beschleunigungssignal mittels der Zählung von Nulldurchläufen des Signals auf erdbebenartige Schläge hin zu untersuchen.

Durch die Empfindlichkeit des Beschleunigungssensor sind jedoch auch in einer absoluten Ruheposition des Gerätes stets Nulldurchläufe des Signals messbar. Daher ist es notwendig, erst Signalwerte, die über einem gewissen Schwellwert liegen, als Nulldurchläufe zu zählen. Dieser Umstand soll in der folgenden Abbildung 4 verdeutlicht werden.

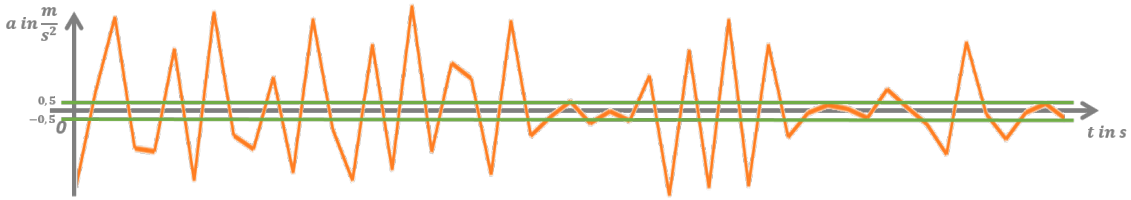


Abbildung 4: Beschleunigungssignal mit Schwellwerten

Die grünen Linien innerhalb des Diagramms sind dabei die definierten Schwellwerte. Hierzu ist in der Implementierung ein Wert von +/- 0,5 $\frac{m}{s^2}$ gewählt worden. Wie im Diagramm ersichtlich, stellt dieser Schwellwert bei wahrnehmbaren Erschütterungen keinerlei Einschränkung dar und filtert zuverlässig kleinere Signalstörungen heraus.
Die eigentliche Erkennung von Erdbeben gestaltet sich nun denkbar einfach und somit auch ressourcenschonend. Für die eigentliche Signalanalysierung werden lediglich zwei Variablen benötigt. Eine Variable legt dabei fest, ob der letzte Signalwert positiv oder negativ war. Die zweite Variable wird bei jedem Signalwechsel von positiv auf negativ erhöht. Betrachtet man diese Auswertemethode anhand des verwendeten Signalverlaufs der vorherigen Abbildungen, so ergibt sich ein Auswerteergebnis, welches in Abbildung 5 dargestellt ist.

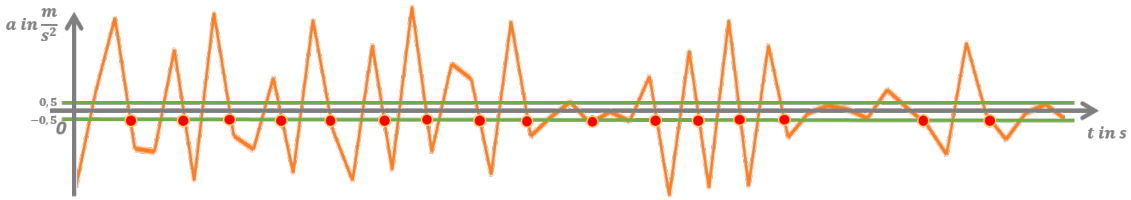


Abbildung 5: Beschleunigungssignal mit Nulldurchläufen analysiert

Jede rote Markierung des Signals steht hierbei für einen Signalwechsel. Wie ersichtlich, ergibt sich somit für jeden einzelnen Ausschlag eine Markierung. Zum Ende des Signals wird dabei auch die Filterung mittels der Schwellwerte deutlich. Hier erreicht ein Signalwechsel diesen Schwellwert nicht und wird deshalb auch nicht mitgezählt.
Zur Erkennung von Erdbeben wird nun in einem bestimmten Zeitrahmen die Anzahl dieser Signalwechsel gezählt. In der momentanen Implementierung wird dazu ein Zeitraum von fünf Sekunden ausgelöst.
Nachfolgend soll die Implementierung in Java erläutert werden. Im Listing 3 sind hierzu die benötigten globalen Variablen aufgeführt.

Listing 3: Globale Variablen der Erbebenerkennung

```
1 private boolean signalGreaterZero = true;
2 private int signalChangeCount = 0;
```

```
3 private long analyseTimeSpan = System.currentTimeMillis();
4 private int accelFreq = 0;
```

Die Variable *signalGreaterZero* speichert jeweils für den letzten Signalwert, ob dieser positiv oder negativ war. Die Variable *signalChangeCount* ist für die Zählung der Signalwechsel verantwortlich. In der Variable *analyseTimeSpan* dient zur Speicherung der Systemzeit, um alle fünf Sekunden eine Auswertung zu machen. Die Variable *acceFreq* dient dem Auswerten der Frequenz des Beschleunigungssensor. Dieser Umstand wird nachfolgend noch genauer erläutert.

Anschließend soll die eigentliche Signalauswertung erklärt werden. Dazu ist zunächst, wie bereits beschrieben, die Zählung der Signalwechsel nötig. Die dafür notwendige Implementierung ist im nachfolgendem Listing 4 aufgeführt.

Listing 4: Implementierung der Erdbebenerkennung

```
1 AccelSample sample = intent.getParcelableExtra(Accelerometer.ACCEL_SAMPLE_KEY);
2     if (sample != null) {
3         if (sample.abs < -0.5 && signalGreaterZero) {
4             signalChangeCount++;
5             signalGreaterZero = false;
6         } else if (sample.abs > 0.5)
7             signalGreaterZero = true;
8     ...
9 }
```

Wie im Auszug ersichtlich, ist die Zählung der Signalwechsel mittels des Beschleunigungssensorwertes aus dem Intent des bereits beschriebenen Broadcast-Receivers implementiert. In der Abfrage wird zunächst überprüft, ob der Wert des Sensors größer ist, als der vorgegebene Schwellwert von +/- 0.5 $\frac{m}{s^2}$. Ist dies der Fall, wird überprüft ob ein Vorzeichenwechsel des Signals stattgefunden hat. Ist dies der Fall, wird beim Vorzeichenwechsel von positiv zu negativ die Variable *signalChangeCounter* erhöht und das neue Vorzeichen mittels der Variable *signalGreaterZero* gespeichert. Nachdem nun die Nulldurchläufe gezählt werden, wird eine Auswertung der gezählten Durchläufe in einem bestimmten Zeitrahmen benötigt. Diese wird im nachfolgenden Listing 5 erläutert.

Listing 5: Analyse der gezählten Nulldurchläufe

```
1 if (System.currentTimeMillis() - analyseTimeSpan> 5000) {
2     double alarmRatio = 0;
3     if(signalChangeCount != 0)
4         alarmRatio = (double)signalChangeCount/(double)accelFreq * 100.0;
5     //Alarmauswertung abhaenig von der Frequenz
6     double alarmFrequRel = accelFreq * 0.01 + 0.5;
7     alarmRatio = alarmRatio * alarmFrequRel;
```

```
8
9       if (alarmRatio > 25)
10          sendAlarmToServer();
11
12      analyseTimeSpan = System.currentTimeMillis();
13      accelFreq = 0;
14      signalChangeCount = 0;
15  }
16  accelFreq++;
```

Die Analyse der gezählten Signalwechsel wird dabei alle fünf Sekunden ausgeführt. Dazu wird die Variable *analyseTimeSpan*, welche den Zeitpunkt der letzten Analyse abspeichert, bei jedem Aufruf des Broadcast-Receivers mit der aktuellen Systemzeit verglichen. Nachdem eine Zeitspanne von über fünf Sekunden vergangen ist, wird die Analyse gestartet.

Wie ersichtlich, reicht es nicht aus, lediglich die Anzahl der gezählten Spitzen zur Auswertung heranzuziehen. Das liegt daran, dass der Sensor innerhalb von Android keine gleichbleibende Frequenz besitzt und sich zudem die Frequenz von Gerät zu Gerät massiv unterscheiden kann. Die Frequenz der Sensoren reicht dabei von 2 Hz bis hin zu 50 Hz. Da bei einer Frequenz von 2 Hz innerhalb von fünf Sekunden lediglich 10 Messwerte verfügbar sind, ist hier eine zuverlässige Erkennung kaum möglich. Für die Auswertung wird deshalb eine Frequenz von 10 Hz als optimal angesehen und Frequenzen, die darüber oder darunter liegen dementsprechend mittels der Berechnung der Variable *alarmFrequRel* angepasst. Die Auswahl der Frequenz von 10 Hz als Referenz begründet sich vor allem dadurch, dass sich während der Implementierung herausgestellt hat, dass die meisten Geräte den Beschleunigungssensor durchschnittlich mit dieser Frequenz aktualisieren. Anhand dieser Referenzfrequenz ist definiert worden, dass ab einem Verhältnis der gemessenen Signalwechsel zu den gesamten gemessenen Werten von größer 25% ein Alarm ausgelöst werden soll. Da bei einer höheren Frequenz die Messwerte eines einzelnen Signalwechsels deutlich zunehmen, muss hier die Gewichtung von gemessenen Signalwechseln zu gemessenen Messwerten erhöht werden. Ist die Frequenz langsamer als 10 Hz, wird die Gewichtung der Signalwechsel dementsprechend reduziert. Ist die Variable *alarmRatio* größer als 25, wird mittels der Methode *sendAlarmToServer* ein Alarm an den Server geschickt. Nach der Auswertung werden alle zählenden Variablen wieder zurückgesetzt und der Zeitpunkt der Analyse in der Variable *analyseTimeSpan* vermerkt, um nach fünf Sekunden erneut eine Analyse starten zu können.

Insgesamt sind die Parameter der Analyse während der Testphase stets angepasst worden und lösen nun bei allen zur Verfügung stehenden Testgeräten mit einer annähernd gleichen Empfindlichkeit einen Alarm aus.

6 Fazit

Zusammengenommen ist in dem Projekt das gewünschte Ergebnis erreicht worden. Alle wesentlichen Anforderungen konnten umgesetzt werden. Das Projekt erforderte eine tiefgehende Auseinandersetzung in die Android und WebService Programmierung unter Java. Ebenso sind die Kompetenzen im Bereich der Teamarbeit bei allen Beteiligen erweitert worden.
Abschließend kann festgestellt werden, dass diese Lösung zur Erdbebenerkennung durchaus sinnvoll eingesetzt werden kann und eines Tages dazu dienen könnte, Sach- und Personenschäden bei einem Erdbeben zu minimieren.