GPS Tacho

Larissa Gärtner: 2081333

Christian Lehr: 2063585

Betreuer: Prof. Dr. Andreas Plaß

21. Juni 14

Studiengang Media Systems (B.Sc.)

Hochschule für angewandte Wissenschaften Hamburg /
Hamburg University of Applied Sciences

Department Medientechnik
Fakultät Design, Medien und Information

Inhalt

1. Projektidee

Für unser Projekt in der Vorlesung Mobile Systeme im Sommersemester 2014 soll ein Messgerät als eine Applikation für iOS oder Android programmiert werden. Die App soll dabei die im Smartphone vorhandenen Sensoren nutzen, welche zuvor in der Vorlesung durch Studentengruppe vorgestellt wurden. Als Sensors/Sensoren stehen Bewegungssensoren (Accelerometer, Gyroskop, Magnetfeld, u.a.), Positionssensoren (GPS, indoor Location, iBeacon, u.a.) und Licht- und Tonsensoren zur Auswahl.

Für die Vorstellung der verschiedenen Sensoren innerhalb der Vorlesung entschieden wir uns durch ein erstes Interesse bedingt für die Bewegungssensoren, also GPS, iBeacon, etc. Durch die von uns durchgeführt Marktanalyse erhielten wir dabei einen ersten Einblick in die Möglichkeit eine App mit Bewegungssensoren zu programmieren. Die Recherche und Vorstellung der Bewegungssensoren Applikationen erhöhte unser Interesse an diesen noch weiter, weshalb wir uns entschieden für unser Projekt einen der Bewegungssensoren zu verwenden. Die Anforderungen an unsere Applikation standen dabei sehr schnell fest. Die App sollte nach Fertigstellung des Projekts für uns im Alltag nutzbar sein. Es sollte sich um eine schlichte App handeln, welche genau so funktioniert, wie man es sich bei der ersten Benutzung intuitiv vorstellt. Dies war uns vor Allem wichtig, da wir ein Projekt abliefern wollten, welches vom ersten Moment an einem festen Konzept folgt und am Ende auch genau so funktioniert und umsetzbar ist, wie wir es uns von Anfang an vorgestellt haben. Nach weiterer Recherche speziell zum Thema Messgeräte und gemeinsamer Absprache entschieden wir uns schließlich dafür eine Applikation zu erstellen, welche das sogenannte globale Postionsbestimmungssystem, also GPS (Global Positioning System) nutzt.

Bei unserer Applikation soll es sich im Detail um eine Art GPS Tacho handeln, welcher bei Aktivierung über einen Start-Button sowohl die Geschwindigkeit in km/h auf einem Tacho anzeigt, als auch die Messdauer in Minuten und den

zurückgelegten Weg in Metern angibt. Des Weiteren soll die Anwendung per Stop-Button unterbrochen und per Reset-Button zurückgesetzt werden können.

Abbildung 1: Verwendeter Tacho

2. Technische Informationen zu GPS

Da wir uns für die Nutzung von GPS zur Erstellung unserer Applikation entschieden haben wollen wir hier nun einmal genauer auf das globale Positionsbestimmungssystem eingehen. Das Global Positioning System (GPS) löste 1985 NNSS, welches ein altes Satellitennavigationssystem der US-Marine war, ab. Entwickelt wurde GPS bereits seit den 1970er Jahren. Es dient als ein globales Navigationssatellitensystem zur Positionsbestimmung. Dabei ist GPS mittlerweile nicht mehr nur für militärische Zwecke interessant, sondern wird heute auch für zivile Zwecke verwendet und das mit einer Ortungsgenauigkeit von oft besser als 10 Metern, welche sich durch weiterführende Differenzmethoden im Bereich des Empfängers auf wenige Zentimeter oder besser steigern lassen.

Um nun mithilfe von GPS seine eigene Position berechnen zu können benötigt man zunächst einmal einen GPS-Empfänger, welcher in unserem Fall ein

Smartphone sein wird. Das Global Positioning System selbst basiert auf Satelliten, welche ihre aktuelle Position und die genaue Uhrzeit mittels codierter Radiosignalen ausstrahle. Der GPS-Empfänger kann unter Berücksichtigung dieser Werte seine eigene Position und seine Geschwindigkeit berechnen. Hierfür reichen dem Empfänger theoretisch die Signale dreier unterschiedlicher Satelliten. Aus den Signallaufzeiten der verwendeten Satelliten kann der seine eigenen Daten berechnen. Die drei mindestens benötigten Satelliten müssen sich jedoch oberhalb ihres Abschaltwinkels befinden, da dadurch die genaue Position und Höhe bestimmt wird. Als Abschaltwinkel bezeichnet man hierbei den kleinst möglichen Höhenwinkel, bei welchem der Empfänger noch die Signale des Satelliten auswerten kann. Praktisch ist die Berechnung jedoch ein wenig komplizierter, da der Empfänger zumeist keine Uhrzeit bereitstellt, welche genau genug wäre, um die Laufzeit der drei Satelliten genug zu messen. Aus diesen Grund wir ein vierten Satelliten in den Prozess miteinbezogen, welche die genau Zeit des Empfängers ermittelt. Zusätzlich zur Positionsbestimmung ist durch GPS auch die Geschwindigkeit des Empfängers berechenbar. Um sicher zu stellen, dass jeder Empfänger zu jeder Zeit mindestens vier Satelliten zur Verfügung hat, umkreisen insgesamt 24 Satelliten die Erde.

Abbildung 2: 24 Satelliten umkreisen die Erde.

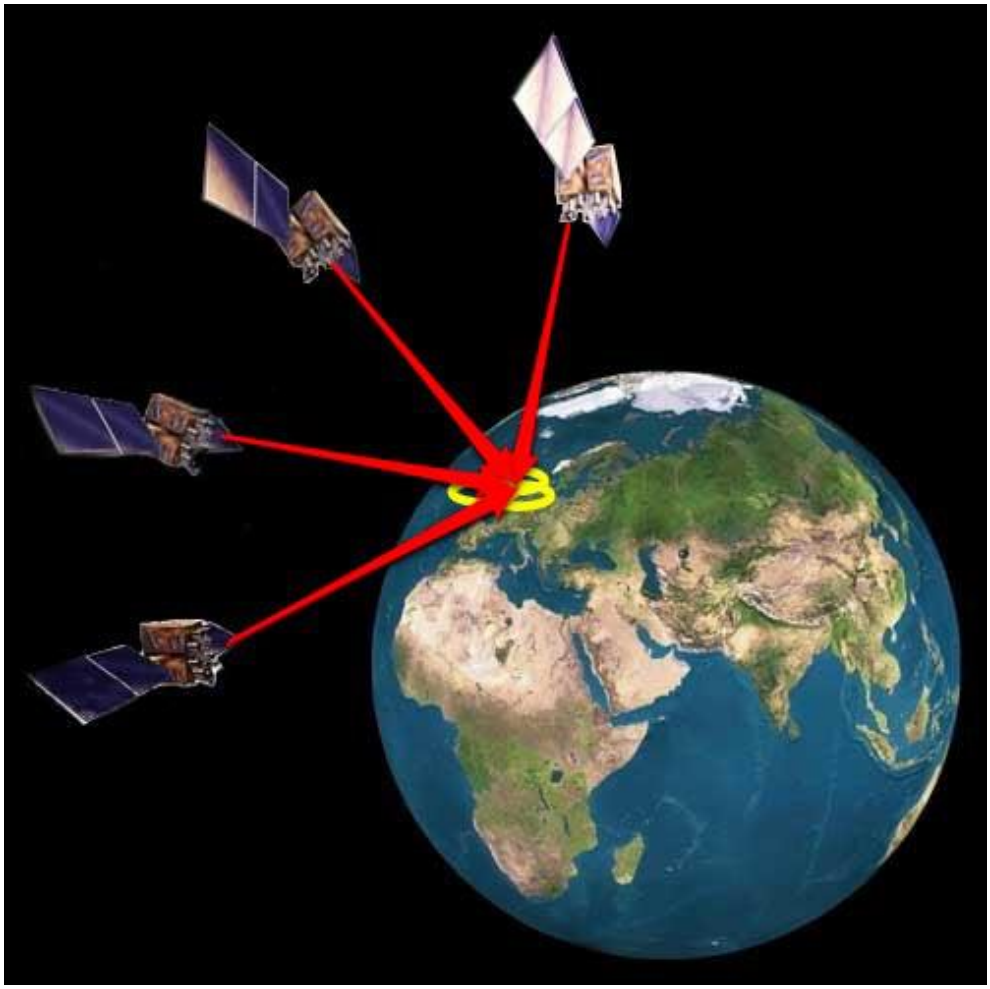
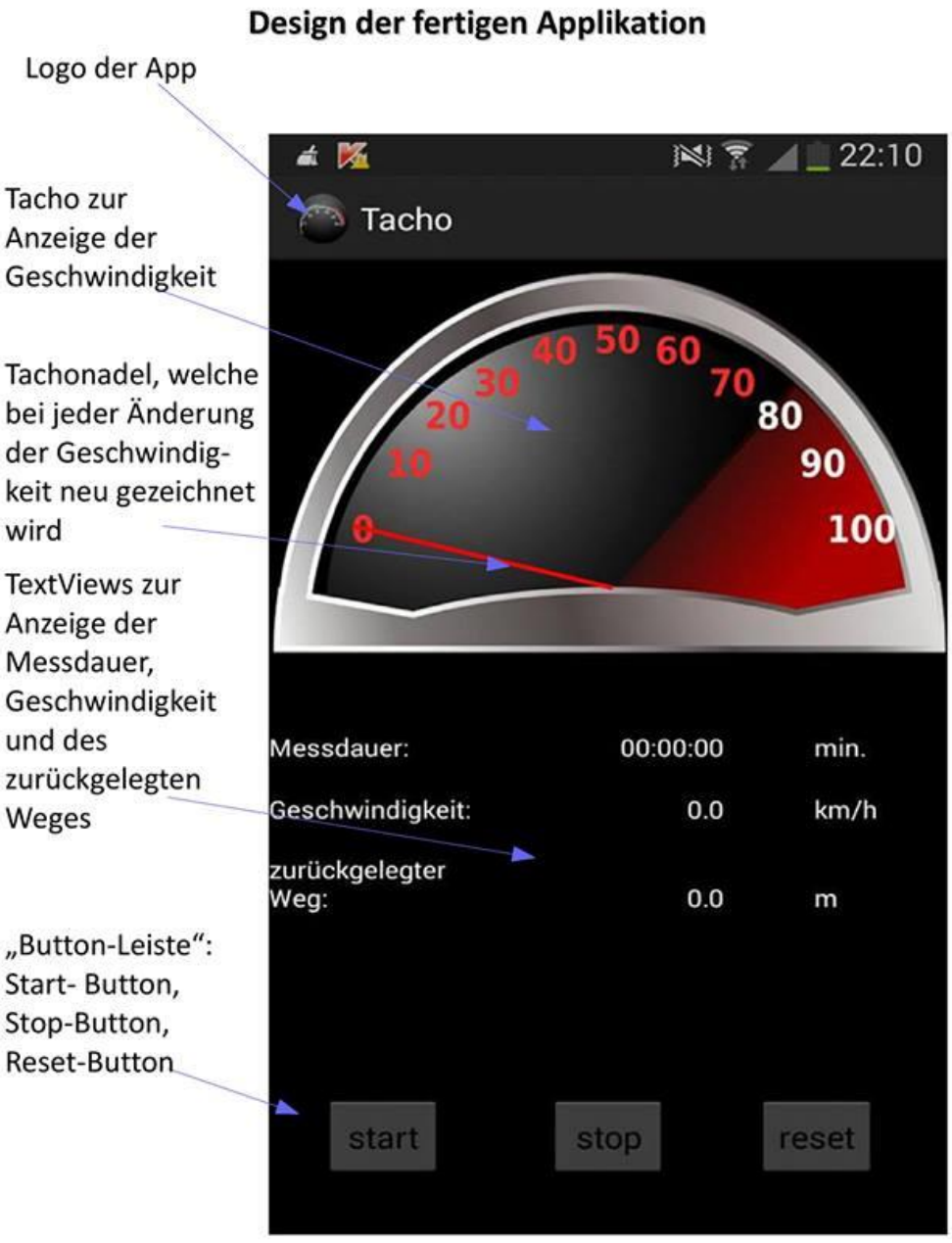


Abbildung 3: Es werden mindestens 4 Satelliten zur Positionsbestimmung benötigt.

3. Applikation (Design)

Zunächst einmal eine Übersicht der fertigen App:



Für das Design der App planten wir die Verwendung von schlichten Elementen, um nicht von den eigentlichen Funktionen abzulenken. Zunächst machten wir uns Gedanken über die Anordnung der genannten Appfunktionen. Das Tacho selbst sollte dabei mittig im oberen Bereich der App positioniert werden und die momentane Geschwindigkeit durch eine Tachonadel angezeigt werden. Die Messdauer und der zurückgelegte Weg sollten in einem TextView unterhalb des Tachos angezeigt werden. Wir entschieden uns hier erneut auch die Geschwindigkeit anzuzeigen zu wollen, da man dort einen genaueren Wert anzeigen kann. Am unteren Rand des Bildschirms planten wir die Buttons zum Starten, Stoppen und Resetten ein. Farblich stellten wir uns das Design dezent aber edel vor und wählten daher rot, weiß, schwarz und grau Töne.
Für das Logo der App wählten wir einen kleinen Tacho als Symbol, um den Nutzen der App schon durch das Symbol zu verdeutlichen. Auch hierfür wähl- ten wir die Abbildung eines Tacho, welche in ähnlichen Farben gehalten war.

Abbildung 4: App-Logo

4. Applikation (Code)

Folgende Dateien sind für das Projekt von Relevanz: attrs.xml, activity_main.xml, MainActivity.xml, TachoView.java, AndroidManifest.xml. Im Folgenden wird auf jede der Dateien eingegangen.

AndroidManifest.xml

Die AndroidManifest.xml-Datei wird automatisch erstellt, wenn man ein neues Android-Projekt erstellt. Diese muss geringfügig verändert werden, damit man GPS und Internet dazu benutzen darf, um an die erforderlichen Daten zu kommen. Dazu fügt man innerhalb der </uses-sdk />- Tags folgenden Befehl ein: <uses-permission android: name="android.permission.ACCESS_FINE_LOCATION" />. Ein weiterer Befehl muss eingefügt werden, damit man die Google Play Library verwenden darf: <meta-data android:name="com.google.android.gms.version" android:value="@inte-

ger/google_play_services_version“/>. Diese wird in diesem Projekt verwendet, da man dadurch genauere Daten für die einzelnen Locations erhält, als mit den Methoden, die standardmäßig von Android zur Verfügung gestellt werden.

Attrs.xml

Diese Datei befindet sich im values-Ordner und sieht wie folgt aus:

```
<resources>
     <declare-styleable name="TachoView">
          <attr name="ersteKoordinate" for-
mat="float" />
          <attr name="zweiteKoordinate" for-
mat="float"></attr>
          <attr name="dritteKoordinate" for-
mat="float"></attr>
          <attr name="vierteKoordinate" for-
mat="float"></attr>
     </declare-styleable>
</resources>
```

Diese Befehle braucht man, weil man hiermit veränderbare Attribute der eigenen View angibt. Dies ist wichtig, damit man später über beispielsweise getter-/setter-Methoden Zugriff auf diese Variablen hat. In diesem Fall sind es 4 Variablen vom Datentypen Float, die wir später brauchen, um unsere Nadel auf dem Tacho richtig zeichnen zu können.

Activity_main.xml

Hier werden die einzelnen Elemente wie Buttons, TextViews, usw. erstellt. Es wird ein RelativeLayout verwendet. Hinzu kommen: eine CustomView namens TachoView, 9 TextViews, 3 für die Zahlenwerte, die restlichen 6 einfach zur näheren Beschreibung der 3 Zahlenwerte, sowie 3 Buttons, darunter ein Startbutton, ein Stopbutton und ein Resetbutton. Die 3 Buttons steuern die App: erst wenn der Startbutton gedrückt wird, beginnt der GPS Tacho Werte zu ermitteln und auszugeben. Wird der Stopbutton gedrückt, so bleibt der Wert der zurückgelegten Strecke sowie der Wert der Stoppuhr erhalten, die Geschwindigkeit hingegen wird auf 0 zurückgesetzt. Beim Drücken des Resetbuttons

werden alle Werte auf 0 zurückgesetzt und somit geht auch die Tachonadel zurück auf die Startposition.

TachoView.java

In dieser Klasse wird der Tacho als Bitmap gezeichnet und die Nadel mit Hilfe der drawLine-Methode. TachoView erbt von der Klasse View.

Im Folgenden wird auf die wichtigsten Methoden eingegangen:

- Konstruktor: Hier wird ein Paint-Objekt mit bestimmten Einstellungen wie Farbe, Dicke für die Tachonadel erstellt. Hier wird ebenfalls auf die Werte der attrs.xml zugegriffen.
- **Void onDraw(Canvas canvas) {}:** hier wird letztendlich das Tacho als Bitmap sowie die Tachonadel gezeichnet.
- **Float getX1() {}:** hier wird einfach der Wert für die erste Koordinate der Tachonadel zurückgegeben. Diese Methode gibt es auch für die anderen Koordinaten X2, Y1, Y2.
- **Void setX1(float newX1) {}:** mit dieser Setter-Methode kann der Wert für die erste Koordinate der Tachonadel gesetzt werden. Auch diese Methode gibt es wieder für jede der vier Koordinaten. Innerhalb dieser Methode wird auch die Methode invalidate() aufgerufen, damit die Nadel auch wirklich an der neuen Position gezeichnet wird.

MainActivity.java

In dieser Klasse wird der LocationListener implementiert, die erforderlichen Daten ermittelt, sowie die TextViews an die Werte angepasst. Im Folgenden die wichtigsten Methoden und Funktionen:

- **getDistance(Location loc1, Location loc2){}:** in dieser Methode wird mit Hilfe der distanceBetween()-Methode die Ditsanz zwischen 2 Locationpunkten ermittelt, in einem Array gespeichert und anschließend der erste Eintrag des Arrays zurückgegeben.

- **Void onCreate(Bundle savedInstanceState){}:** Hier wird ein LocationClient erzeugt und angegeben wie oft der Client die Position messen soll. Weiterhin werden die Elemente des Layouts wie Buttons und TextViews instantiiert. Ebenfalls werden die 3 Buttons mit einem OnClickListener versehen und zwar mit Hilfe einer anonymen OnClickListener-Klasse. Die groben Funktionen der 3 Buttons wurden ja bereits in der activity_main.xml beschrieben.
- unser GPS Tacho hat ebenfalls eine Stoppuhr implementiert. Dies wird über ein Runnable-Objekt und einen Handler bewerkstelligt. Interessant sind dabei die Aufrufe SystemClock.uptimeMillis(), myHandler.removeCallbacks(myRunnable) und myHandler.postDelayed(this,0).
- **onLocationChanged(Location location){}:** sobald eine Location über mLocationClient.getLastLocation() ermittelt wurde, wird ebenfalls ein Wert für die Geschwindigkeit mit dem Aufruf location.getSpeed() ermittelt. Anschließend wird über if-Abfragen je nach Höhe des Geschwindigkeitswertes eine Nadel im Tacho gezeichnet und zwar so, dass in etwa die richtige Geschwindigkeit angezeigt wird. Dies geschieht über setter-Methoden, in denen 2 Koordinaten für die Nadel gesetzt werden, die anderen 2 Koordinaten (man braucht insgesamt 4, um eine Linie zeichnen zu können) bleiben immer gleich. Folgendes Beispiel demonstriert das Setzen der Koordinaten für die Nadel an der Position „0“, also der Startposition beim Tacho:

```
if (speed >= 0 && speed <= 5) {
     myView.setX1(100);
     myView.setY1(300);
}
```

Dabei wird der Geschwindigkeitswert wie oben beschrieben in 5er-Schritten mit Hilfe von if-Abfragen bearbeitet, also von 0-5 km/h, 5-10 km/h, 10-15 km/h, ….., <105 km/h, sodass die Nadel an 22 verschiedenen Positionen gezeichnet werden kann, je nach gemessener Geschwindigkeit. Anschließend wird noch mit Hilfe der Methode getDistance(Location loc1, Location loc2)

der zurückgelegte Weg ermittelt und zur Variablen „d“ vom Datentypen float dazugerechnet. Da dies bei jedem Aufruf von onLocationChanged passiert, gibt „d“ somit den Weg an, der insgesamt zurückgelegt wurde.

```
protected void onStart() {
        super.onStart();
        // Connect the client.
        mLocationClient.connect();
}
```

In dieser Methode verbinden wir uns mit dem LocationClient.

```
protected void onStop() {
// Disconnecting the client invalidates it.
        super.onStop();
        mLocationClient.disconnect();
  }
```

Und in dieser Methode meldet man den LocationClient ab.

5. Optimierung

Nach der Fertigstellung der Applikation machten wir uns Gedanken darüber, wie man diese in Zukunft durch weitere Bearbeitung noch verbessern bzw. verschönern könnte. Als erste fielen uns hierbei die Buttons auf, welche man durch eher geringen Aufwand in einem großen Maße verschönern könnte, da diese durchaus ansehnlicher aussehen könnten. Des Weiteren wäre eine schönere Darstellung der Tachonadel wünschenswert. Da uns das Erstellen dieser jedoch starke Probleme während der Erstellung der Applikation bescherte konnten wir dies nicht mehr realisieren. Auch die Darstellung der Werte unterhalb des Tachos könnten durch eine entsprechende Überarbeitung ansehnlicher gestaltet werden. Zu guter Letzt würden wir in weiterer Bearbeitung gerne einen größeren Tacho einbauen, d.h. ein Tacho, welches nicht bei 100 km/h endet und in Form eines Kreises dargestellt wird ähnlich der Tachos in Autos.

6. Quellen

Global Positionin System. (n.d.). In *Wikipedia*. Abgerufen 18. Juni 2014 von http://de.wikipedia.org/wiki/Global_Positioning_System

Satellitenbild.(n.d.). Abgerufen 18. Juni 2014 von http://www8.garmin.com/graphics/24satellite.jpg

Satellitenbild.(n.d.). Abgerufen 18. Juni 2014 von http://www.pocketgps-world.com/reviews/howgpsworks/gps2.jpg

7. Abbildungsverzeichnis