Vorgehen Applikationsentwicklung

# Aufbau der Applikation

Nachfolgend wird der Aufbau von TI SensorTag zusammen mit unserem selbst entwickelten Board als Sensor benannt und die Android Applikation auf dem Smartphone wird einfach als Applikation benannt.

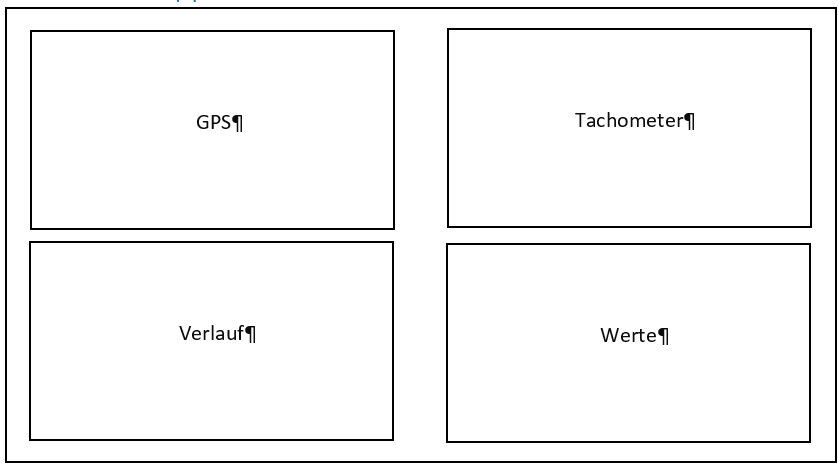


Abbildung 1: Aufteilung des Bildschirms

Der Aufbau der Applikation soll Anfangs in vier Teile des Bildschirms geteilt werden.

Im Bereich GSP soll eine Karte angezeigt werden, ausserdem soll es möglich sein die Route aufzuzeichnen und abzuspeichern.

Der Verlauf zeigt die Entwicklung der Geschwindigkeit über einen gewissen Zeitraum, wobei es wählbar sein soll welche Daten angezeigt werden.

Die Geschwindigkeit soll anschaulich in einem Tachometer angezeigt werden und die Tachonadel soll im besten Fall animiert sein.

Im Bereich der Werte sollen die aktuellen Werte in Zahlen und den entsprechenden Einheiten dargestellt werden.

Schnell wurde klar, dass dieser Aufbau wenig Sinn macht, da die Bereiche auf einem Smartphone viel zu klein wären und nicht mehr leserlich sein würden. Darum wurde ein neues Konzept erarbeitet, welches in der Abbildung x ersichtlich ist.

## Home-Screen

Als erstes soll der Benutzer den sogenannten Home-Screen sehen, hier werden der Tachometer, die Werte und ein paar Buttons angezeigt. Je nach Anzahl der Funktionen sollen neue Buttons implementiert werden. So soll beispielsweise für die Sensorwahl ein Button vorhanden sein. Während der Entwicklung der Applikation wurde die Idee von einer GPS-Karte und einem Verlauf ebenfalls verworfen, da diese Funktionen aus zeitlichen Gründen nicht mehr realisierbar waren. Jedoch wäre der Home-Screen in der Lage mehr Buttons aufzunehmen, es muss nur darauf geachtet werden, dass die Anzeige der Werte nicht zu gross wird.

## Sensorwahl

Schnell wurde klar, dass die Auswahl eines spezifischen Sensors wichtig werden würde, für den Fall, dass einmal mehrere Sensoren vorhanden wären. Ein Beispiel, würde unsere Applikation bei einem Fahrradrennen eingesetzt werden, so müsste die Applikation auf dem Smartphone mit dem Sensor am Fahrrad gepaart werden. Ansonsten würde man die Daten von allen Sensoren empfangen und die Anzeige würde nicht die Daten des eigenen Sensors anzeigen.

## Einheiten und Einstellungen

Es kam ebenfalls der Wunsch auf, die Einheiten der Werte, einstellbar zu machen, da die Applikation bestenfalls auch in anderen Ländern verwendet werden soll. Also wäre es optimal, wenn die Geschwindigkeit beispielsweise auch in Miles per Hour dargestellt werden könnte. Vielleicht müsste auch eine Einstellungsmöglichkeit bereitgestellt werden, damit die Temperatur kalibriert werden kann.

# BLE-Kommunikation

Der wichtigste Teil der Applikation ist die Bluetooth-Kommunikation, hier werden die Daten vom Sensor empfangen. Die Daten müssen nach dem Empfangen gefiltert werden, damit sichergestellt wird, das die empfangenen Daten von einem unserer Sensoren kommen. Sobald sichergestellt ist, das die empfangenen Daten zu unserem Sensor gehört, müssen diese umgerechnet werden.

## Inbetriebnahme Code PA

Als erster Schritt wurde der Code der vorangegangenen Arbeit analysiert und die wichtigen Teile zur Bluetoothkommunikation übernommen. Es wurden folgende Funktionen aus dem Code übernommen: onStart, onActivityResult, scanLeDevice, BluetoothAdapter.LeScanCallback.

Die Funktion onStart wurde in BLE\_init umbenannt und der Mechanismus zum Wachhalten des Geräts wurde entfernt. Der Mechanismus wurde entfernt, da dieser nicht mehr funktionierte. Ebenfalls wurde die Funktion in die Funktion onCreate eingefügt, um den Code übersichtlicher zu gestalten und da diese Funktion nur einmalig aufgerufen werden musste.

Die Methoden onActivityResult und scanLeDevice wurden komplett übernommen.

Die Callback-Funktion des Bluetoothadapters BluetoothAdapter.LeScanCallback wurde strukturell übernommen, jedoch wurde die Codierung zur Verarbeitung der Daten entfernt.

## Filter einbauen

Der nächste logische Schritt war, dass die empfangenen Daten gefiltert werden mussten. Dafür wurde folgende Struktur definiert, damit festgestellt werden kann, ob die Daten von einem unserer Sensoren versendet wurden.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Length | Type | UUID | Paket-ID | Speed | Pressure | Temperature | Huminity | Checksumme |
| 23 | 0x03 | 0xDEBA | 2 Byte | 4 Byte | 4 Byte | 4 Byte | 4 Byte | 2 Byte |

Aufgrund dieser Struktur können alle empfangenen Daten, welche nicht die definierte Länge, Typ und UUID besitzen, ignoriert werden.

Wenn sichergestellt ist, dass die Daten von unserem Sensor stammen, musste noch die Adresse vom Sender gespeichert werden. Alle Adressen werden in einer Liste gespeichert, vor dem Speichern wird jedoch noch geprüft, ob die Adresse bereits in der Liste ist, falls ja wird die Adresse nicht noch einmal gespeichert.

## Berechnung der Daten

Der Sensor sendet Werte, wie den Zeitunterschied zwischen zwei Magnetdurchläufen, dem Luftdruck, der Temperatur und die relative Luftfeuchtigkeit. Diese Daten müssen erst noch in die realen Werte umgerechnet werden, da alle Werte in mehreren Bytes verteilt vorliegen.

Der Zeitunterschied zwischen zwei Magnetdurchläufen wird in vier Bytes dargestellt. Die ersten beiden Bytes enthalten die Anzahl Sekunden, es muss also das erste Byte um acht Bit geschoben werden und anschliessen muss das zweite Byte noch addiert werden. Folgende Formel zeigt die Umrechnung von zwei Bytewerten in eine reelle Zahl:

Die erhaltene Zahl muss noch mit einem Typecast in den Datentyp float umgewandelt werden.

Das dritte und vierte Byte des übermittelten Zeitunterschieds enthält die Sekundenbruchteile. Folgende Formel zeigt, wie die Bytes verrechnet werden müssen, um die korrekten Werte zu erhalten:

Der erhaltene Sekundenbruchteil kann nun zu den ganzen Sekunden dazu addiert werden, um die ganze Zeit zwischen zwei Magnetdurchläufen zu erhalten.

Ein Beispiel, wird also der hexadezimale Wert 0x00014000 empfangen sind das 1.25 Sekunden zwischen zwei Magnetdurchläufen.

Um die Geschwindigkeit zu erhalten muss der Radumfang durch die erhaltene Zeit geteilt werden. Jedoch befinden sich an unserem Rad zwei Magnete, deswegen darf nur der halbe Radumfang verrechnet werden. Ausserdem ist das Resultat in m/s und muss deshalb noch in km/h umgerechnet werden, was nichts anderes als eine Multiplikation mit dem Faktor 3.6 ist.

Der Druck wird in Pa übertragen, hier müssen die verschiedenen Bytes miteinander addiert werden und in hPa umgerechnet werden. Es werden jedoch nur die letzten drei Bytes mit Werten befüllt sein, das erste Byte kann ignoriert werden. Folgende Formel zeigt die Umrechnung der Bytes in eine float Zahl.

Die barometrische Höhenformel zeigt die Abhängigkeit des Druckes von der aktuellen Höhe.

(Referenz: Buch "Mathematik für Ingenieure und Naturwissenschaftler Band 1, 12. Auflage, Lothar Papula, Seite 284)

Durch Umformen ergibt sich die folgende Formel, womit sich die Höhe aus dem Druck berechnen lässt.

Die Temperatur wird in zwei Bytes übertragen. jedoch wird die Temperatur nicht in °C übertragen, sondern in Tausendsteln. Das bedeutet die Temperatur, welche empfangen wird, muss zusammengesetzt und mit 1000 dividiert werden. Die Formel sieht folgendermassen aus:

Der letzte Wert, welcher vom Sensor versendet wird, ist die relative Luftfeuchtigkeit. Die Formel zur Berechnung der relativen Luftfeuchtigkeit sieht folgendermassen aus:

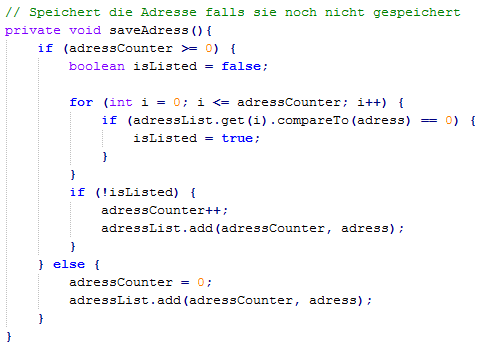
# Einstellungsmöglichkeiten

Als nächstes wurde die Aufgabe in Angriff genommen, die Applikation als User einstellen zu können. Es musste genau überlegt werden, welche Einstellungen ein User machen darf und welche dem Programmierer, bzw. der Applikation überlassen werden.

## Adressauswahl

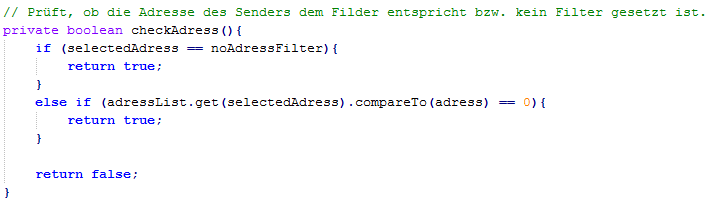
Die Adressauswahl, oder auch Sensorwahl, wurde bald ein Bedürfnis, da der User die Applikation mit seinem eigenen Sensor paaren können sollte. Es sollte sichergestellt werden, dass die Applikation nach der Sensorwahl nur noch Daten von dem gewählten Sensor anzeigt und die Daten der anderen Sensoren ignoriert werden. Die Adressen von Sensoren in der Nähe sollten aber weiterhin gespeichert werden, damit man gegebenenfalls auch einen anderen Sensor auswählen konnte.

Immer wenn Daten empfangen werden und die Struktur mit der definierten Struktur übereinstimmen soll die Adresse gespeichert werden. Dafür wurde folgende Funktion entwickelt:



Der adressCounter wird mit dem Wert -1 initialisiert, werden die ersten Daten mit der vorgegeben Struktur empfangen wird die Funktion automatisch die Adresse in der adressList an der Stelle 0 speichern. Werten erneut Daten von einem unserer Sensoren empfangen wird zuerst geprüft, ob die Adresse bereits ein Teil der adressList ist, falls die Adresse noch nicht in der Liste enthalten ist, wird der Index, bzw. der adressCounter, erhöht und die Adresse in der Liste gespeichert.

Sobald eine Adresse ausgewählt wurde müssen die zukünftigen Datenpackete überprüft werden, ob sie vom ausgewählten Sensor stammen. Die Funktion checkAdress überprüft, ob die ausgewählte Adresse der Adresse des Sensors entspricht, welche die neuen Daten gesendet hat. Sollte keine Adresse ausgewählt sein, so wird dies genauso behandelt, als ob der Sensor die richtige Adresse hat.



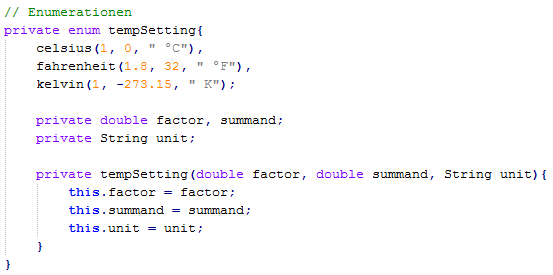
Der User muss noch die Möglichkeit haben eine Adresse auszuwählen, dafür wird eine neue Activity erstellt. Die neue Activity ist eine List-Activity, das bedeutet es wird eine Liste von möglichen Adressen angezeigt, welche mit einem Klick auf die gewünschte Adresse ausgewählt wird. Anschliessend wird die ausgewählte Adresse an die Haupt-Activity zurückgegeben. In Abbildung 2 ist die List-Activity zu sehen, welche eine Adresse zur Auswahl hat. Damit die ausgewählte Adresse von der Haupt-Activity überhaupt empfangen werden kann musste die List-Activity mit den auszuwählenden Adressen mit folgendem Befehl gestartet werden: startActivityForResult. Die gestartete Activity kann somit ein Resultat, also die ausgewählte Adresse zurückgeben und die Haupt-Activity kann diese in der Funktion onActivityResult empfangen und abspeichern. Die genaue Ausführung der Übergabe der Adressen, sowie die Rückgabe der ausgewählten Adresse kann dem Quellcode entnommen werden.



Abbildung 2: Adressauswahl mit einer Adresse

## Einheiten

Die Einheiten der empfangenen Daten sollten ebenfalls einstellbar gestaltet werden. Für diesen Zweck wurden mehrere Enumeration definiert, die Enumeration für die Temperatur sieht folgendermassen aus:



Es wurde der Enumeration drei Elemente hinzugefügt, welche alle einen Faktor, einen Offset und einen String mit der geschriebenen Einheit enthalten. Anhand dieser Enumeration kann der Faktor und der Offset einfach zugänglich gemacht werden.

(Referenz Temperaturumrechnung: <http://www.umrechnung.org/masseinheiten-temperatur-celsius-fahrenheit-kelvin/celsius-fahrenheit-umrechnung.htm> (Konsultation am 22.04.16))

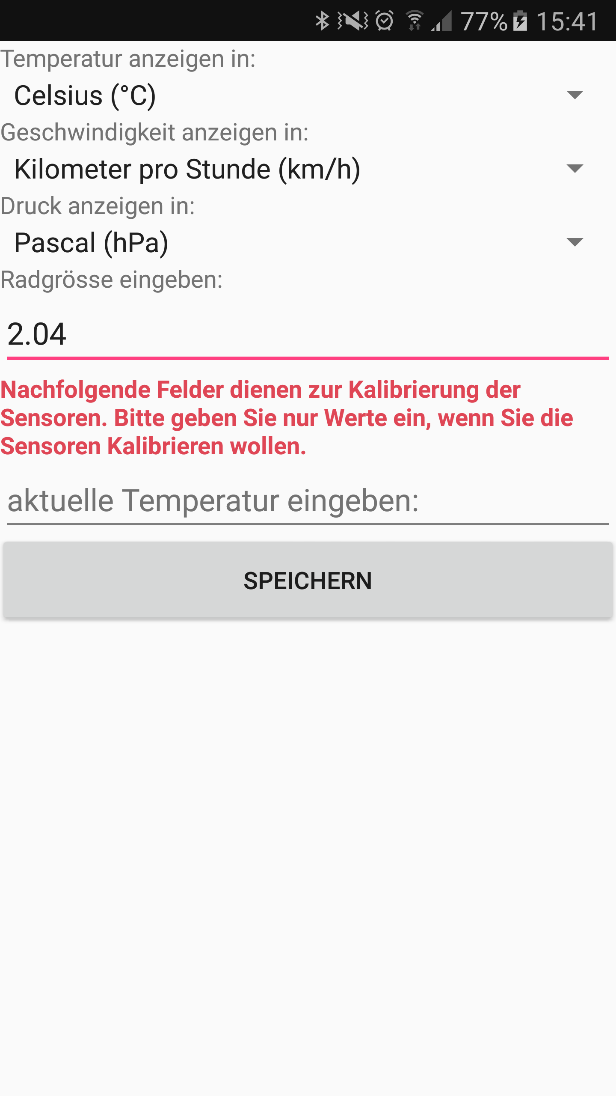
(Referenz Druckumrechnung: <http://www.dolder-ing.ch/wissen/Einheiten/Tab_Druckumr/Tab-Druckumrechn-Gebrf.htm> (Konsultation am 22.04.16))

(Referenz Geschwindkeitsumrechnung: <http://www.convertworld.com/de/geschwindigkeit/knoten.html> (Konsultation am 22.04.16))

Die Enumerationen wurden anschliessend verwendet um die aktuelle Einstellung der Einheiten abzuspeichern. Da die Einstellung der Einheiten nun gespeichert werden konnte, musste diese Einstellung dem User zugänglich gemacht werden.

Eine neue Activity wurde codiert, welche drei Spinner zur Auswahl der Einheiten, einen EditText für die Eingabe des Radumfangs und ein EditText für die Kalibrierung der Temperatur zur Verfügung stellt. Die eingestellten Einheiten und die eingetragenen Werte werden über den Speichern-Button an die Haupt-Activity zurückgegeben. In der onActivityResult Methode in der Haupt-Activity werden die aktuellen Einstellungen empfangen und abgespeichert und anschliessend zur Darstellung der Werte verwendet. Erst bei der Anzeige der Werte, werden die eingestellten Faktoren und Summanden der Enumerationen verwendet.

Abschliessend wurde die Activity zur Einstellung der Einheiten noch dahingehend verbessert, dass die eingestellten Einheiten und die eingetragenen Werte, bei erneutem Aufrufen der Activity dargestellt werden. Wird die Activity zur Einstellung der Einheiten gestartet, wird ihr die aktuelle Einstellung übergeben und die Spinner und EditText zeigen, die zuvor abgespeicherten Werte an. Ein Beispiel, wenn man in der Activity den Spinner für die Temperatur auf "Fahrenheit (°F)" einstellt und diese Einstellung abspeichert. Die Activity dann erneut über einen Druck auf den Button startet, wird beim Spinner für die Temperatur die Einstellung "Fahrenheit (°F)" angezeigt, da diese Einstellung zuvor abgespeichert wurde.



## Kalibrierung

Während der Entwicklung wurde festgestellt, dass der Wert der Temperatur von der Referenztemperatur abweicht. Aus diesem Grund wurde eine Möglichkeit zur Kalibrierung der Temperatur geschaffen.

Die Kalibrierungsmöglichkeit befindet sich in der gleichen Activity, wie die Einstellungen der Einheiten und der Eingabemöglichkeit für den Radumfang. Wird bei dem Textfeld ein Wert eingegeben, so wird diese Temperatur an die Haupt-Activity zurückgegeben. Die Referenztemperatur wird gespeichert und beim Empfang der nächsten Daten, wird die automatisch ein Offset berechnet. Dieser Offset fliesst automatisch in die Berechnung der Temperatur ein.

# Animierter Tachometer

Eine interessante Aufgabe war die Realisierung eines animierten Tachometers, da diese Funktion dem User der Applikation sofort ins Auge stechen würde. Diese Funktion würde die ansonsten relativ triste Aufmachung der Applikation aufwerten. Es sollte ein Tachometer entwickelt werden, welches eine animierte Tachonadel hat, die die aktuelle Geschwindigkeit anzeigt.

## Funktionsweise

Bereits in der PA von Manuel König wurde mit der Anzeige von Bitmaps experimentiert. Es lag nahe, diese Erfahrung zu nutzen und für dieses Problem eine möglichst einfache Lösung zu generieren. In Android ist es möglich ein Bitmap in ein Zahlenarray zu laden, dieses Array anzupassen und schlussendlich das Array wieder in ein Bitmap umzuwandeln.

Die Funktion drawTacho liest die Pixel eines Bildes eines Tachometers ohne Tachonadel ein, dreht es und fügt die rote Tachonadel direkt in das Bild ein.

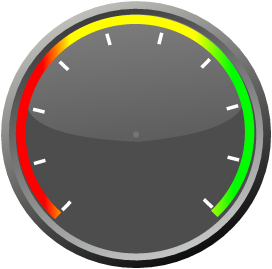


Abbildung 3: Tachometer ohne Tachonadel

Das Bild konnte mit folgender Funktion eingelesen werden:



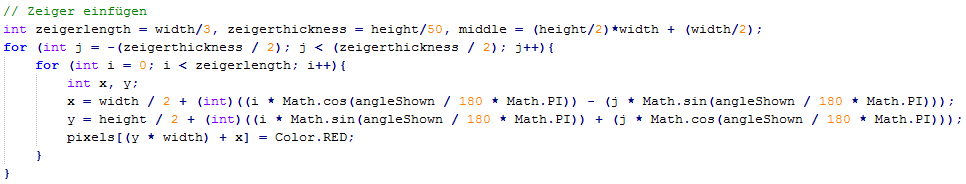
Die Bitmap wird in ein eindimensionales Integer-Array eingelesen, die Werte im Array stellen den Farbwert des jeweiligen Pixels dar. Diese Farbwerte können nun verändert werden. Als erstes wurde das ganze Bild gespiegelt, da es noch falsch herum ist, heisst der rote Bereich des Tachometers soll auf der rechten Seite angezeigt werden.

Nach dem Spiegeln des Bildes musste noch die Tachonadel direkt in das Integer-Array eingefügt werden. Die genaue Vorgehensweise wird im nachfolgenden Punkt genauer beschrieben.

Aus dem bearbeiten Integer-Array konnte anschliessend eine Bitmap erstellt werden, welches von der Funktion zurückgegeben wurde. Wichtig ist dabei, dass die richtige Konfiguration ausgewählt wurde, da ansonsten das Bitmap falsch oder gar nicht angezeigt werden konnte.



## Mathematik der Tachonadel



Die Tachonadel wurde in mehreren Schritten animiert, erst wurde ein Strich mit der Breite von einem Pixel animiert. Die Tachonadel kann als ein Vektor angesehen werden, der im Ursprung gedreht und anschliessend verschoben wurde. Die X- bzw. Y-Koordinaten im Bild konnten folgendermassen berechnet werden:

Anschliessend sollte die Tachonadel dicker werden, da ein Strich mit der Breite von nur einem Pixel nicht leicht auf einem hochauflösenden Display, wie dem Display des Samsung Galaxy S7, zu erkennen ist.

Die Betrachtungsweise, dass die Tachonadel ein Vektor ist musste angepasst werden. Jeder einzelne Pixel hatte einen Ortsvektor der angepasst werden musste. Nachfolgende Formel zur Berechnung der Koordinaten nach einer Drehung des Vektors konnte angewendet werden.

*(Referenz:* [*http://www.mathebibel.de/drehmatrix*](http://www.mathebibel.de/drehmatrix) *(Konsultierung am 22.05.16))*

Diese Formeln mussten noch mit der Verschiebung in den Mittelpunkt des Tachometers ergänzt werden. Die Koordinaten jedes Pixels der Tachonadel konnten nun berechnet werden und wurden anschliessend mit dem Farbwert für Rot in dem Integer-Array überschrieben.

