Erzeugen einfacher OpenStreetMap-Karten

Jens Pönisch*
TU Chemnitz

OpenStreetMap

Das 2004 von Steve Coast gegründete Projekt hat sich zum Ziel gesetzt, geographische Daten wie Landschaftsnutzung, Besiedelungs- und Infrastruktur umfassend und unter einer freien Lizenz (ODBL) in Form einer Datenbank zur Verfügung zu stellen.

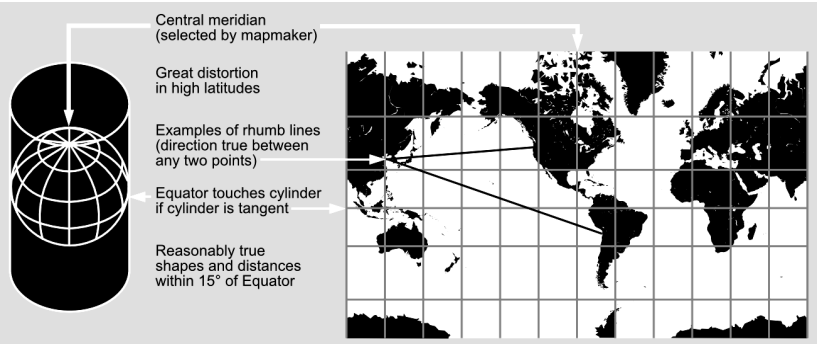
Die sicher wichtigste Nutzungsform ist die Erzeugung von geographischen Karten. So bietet das Projekt selbst einige Online-Karten in verschiedenen Stilen an. Aus diesen bereits aufbereiteten Daten sollen mit möglichst einfachen Mitteln eigene Karten für verschiedene Anwendungsszenarien erzeugt werden.

Nicht eingegangen wird hier auf *routingfähige* Karten, wie sie für Navigationssysteme benötigt werden.

Sphärische Mercatorprojektion und Slippy Map

Da die Erdoberfläche näherungsweise ein Ellipsoid ist, Karten aber flach sind, ist eine *Projektion* von dieser Oberfläche in die Kartenebene erforderlich. Sollen die erzeugten Karten zur Navigation dienen, ist es sinnvoll, wenn Himmelsrichtungen und Winkel an Kreuzungen unverändert bleiben, was mithilfe der *Mercatorprojektion* möglich ist.

Die Erde wird dazu auf einen übergestülpten Zylinder, der in Richtung der Erdachse ausgerichtet ist, projiziert. Anschließend wird der Zylinder an der Datumslinie (180° Ost = 180° West) aufgeschnitten und geeignet zu einem Quadrat verzerrt.



*poenisch@isym.tu-chemnitz.de

Mercatorprojektion (Quelle: Wikimedia Commons)

Zur Beschleunigung der Rechnung wird diese zur *Sphärischen Mercatorprojektion* vereinfacht, die von einer exakten Kugelgestalt der Erde ausgeht und dabei minimale Abweichungen in Kauf nimmt.

Die Berechnung erfolgt nach folgender Formel:

$$
\begin{aligned}
x &= s \cdot \lambda \\
y &= s \cdot \ln\left(\tan\left(\frac{\pi}{4} + \frac{\varphi}{2}\right)\right) \\
&= s \cdot \ln\left(\tan\varphi + \frac{1}{\cos\varphi}\right)
\end{aligned}
$$

λ – Längengrad in Radiant

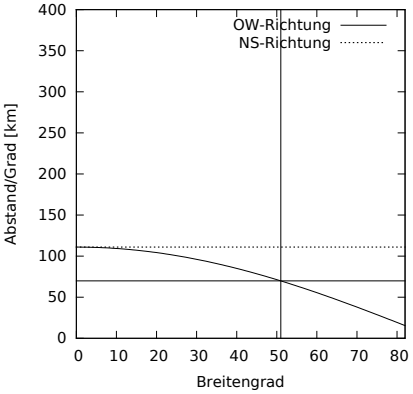
φ – Breitengrad in Radiant

s – Skalierungsfaktor

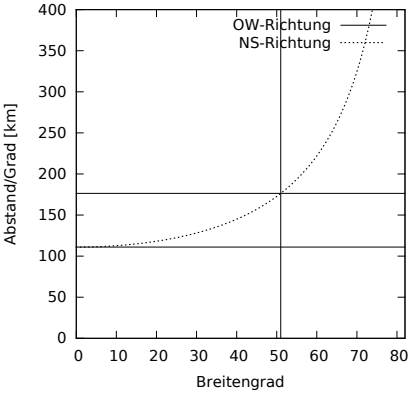
Die Längen- und Breitengrade (im WGS84-Referenzsystem) müssen zunächst in das Bogenmaß umgerechnet werden. Östliche Längen und nördliche Breiten haben dabei ein positives, westliche Längen und südliche Breiten ein negatives Vorzeichen:

$$\text{Radiant} = \frac{\pi}{180^\circ} \cdot \text{Grad}$$

Das Ergebnis dieser Projektion ist ein Quadrat, das die Erdoberfläche bis zum 82-ten Breitengrad nach Norden und Süden abbildet. Abstände werden dabei umso mehr vergrößert, je weiter man sich vom Äquator entfernt, da die Längengrade in der Projektion parallel verlaufen und dafür der Abstand der Breitengrade vergrößert wird. Auf dem 51-ten Breitengrad (Höhe Chemnitz) ist dieser Verzerrungsfaktor etwa 1,6.



Tatsächlicher Abstand je Grad



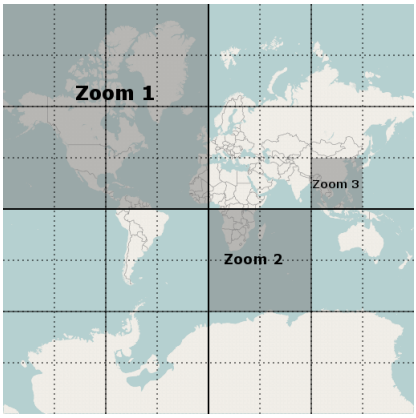
Abstand je Grad in Mercatorprojektion

Dieser Umstand führt bei großflächigen Karten zu einer verfälschten Darstellung von Flächenverhältnissen, bei regionalen Karten muss er lediglich bei der Angabe eines Kilometerrasters oder einer Entfernungsskala berücksichtigt werden.

Für die Kartendarstellung wird dieses Quadrat je nach Zoomstufe umskaliert und in quadratische Kacheln der Größe von 256×256 Pixel zerlegt. Diese werden, beginnend in der linken oberen Ecke, in x- und y-Richtung durchnumeriert. Eine bestimmte Kachel wird also durch das Tripel *(Zoomstufe, x-Koordinate, y-Koordinate)* charakterisiert.

Die Umrechnung von geographischen in Kachelkoordinaten erfolgt durch Anpassung obiger Formel:

$$x = \left\lfloor 2^z \cdot \frac{\lambda + \pi}{2 \cdot \pi} \right\rfloor \qquad y = \left\lfloor 2^z \cdot \frac{\pi - \ln(\tan\varphi + \frac{1}{\cos\varphi})}{2 \cdot \pi} \right\rfloor$$



Konstruktion der Slippy-Map

Dieses Konzept wurde von *Google Maps* übernommen und wird im *OpenStreetMap*-Umfeld als *Slippy Map* bezeichnet. Mit jeder höheren Zoomstufe verdoppelt sich dabei die Zahl der Kacheln in waagerechter und senkrechter Richtung, es wird also insgesamt die vierfache Kachelzahl benötigt. Üblicherweise werden die Zoomstufen bis 16 oder 18 bereitgestellt.

Für die Darstellung regionaler Karten sind besonders folgende Zoomstufen interessant. Die Größenangaben beziehen sich wieder auf den 51-ten Breitengrad.

Zoom	Kachellänge/-breite [km]	Maßstab bei 4 cm Druckbreite
11	12,5	1:300 000
12	6	1:150 000
13	3	1:75 000
14	1,5	1:38 000
15	0,8	1:19 000

Karten für den Ausdruck

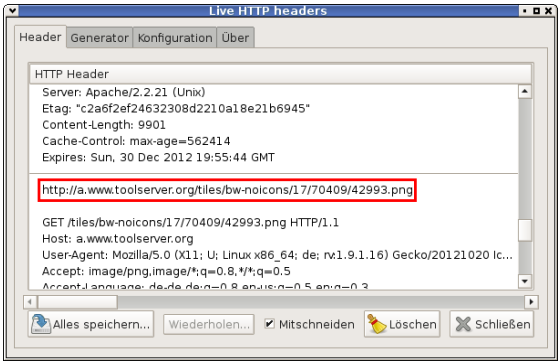
Um eine Karte für den Ausdruck zusammenzustellen, müssen die Kacheln für die gewünschte Zoomstufe besorgt und diese zu einem Bild zusammengestellt werden. Einige Online-Karten bieten diese Funktionalität über ihr Web-Interface selbst an. Fehlt diese Funktion, können diese Schritte manuell durchgeführt werden.

Die von Online-Karten verwendeten Kacheln sind praktisch immer über eine *URL* erreichbar, die jedoch nicht unbedingt mit der *URL* der Kartenanwendung übereinstimmt. Um die benötigten Daten zu ermitteln, kann man entweder den *JavaScript*-Quelltext der Kartenanwendung oder die von ihr ausgelösten *HTML*-Requests analysieren.

Die zweite Variante lässt sich bequem mithilfe des Firefox-Add-ons *Live HTTP headers* [4] realisieren, wobei die Kachel-Requests üblicherweise den Aufbau

http://pfad/*zoom*/*x*/*y*.png

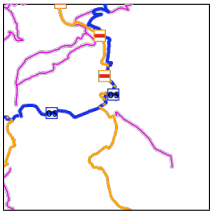
haben und so leicht zu erkennen sind.



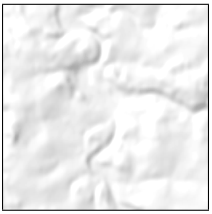
Kachel-Anfrage in Live HTTP headers

Es gibt Basiskacheln, die die eigentliche Kartendarstellung enthalten, sowie häufig *Overlays* mit Zusatzinformationen wie Wanderwegen, Höhenschattierungen und -linien. Letzere sind weitestgehend transparent und können die Basiskacheln überlagern.

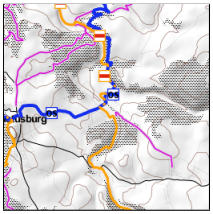
Mapnik-Basiskarte



Overlay Wanderwege (Lonvia)



Overlay Höhen (Hike-and-Bikemap)



Basiskarte mit Overlays kombiniert

Nach Festlegung der Zoomstufe können die x- und y-Werte der benötigten Kacheln durch obige Umrechnung aus den geographischen Koordinaten der darzustellenden Region berechnet werden.

Der Download kann nun mit *curl* oder *wget* erfolgen, das Zusammensetzen der Einzelkacheln zu einer Gesamtgraphik wird durch die Graphikbibliotheken vieler Scriptsprachen unterstützt. Eine mögliche Vorlage ist das Python-Script `bigmap.py` des Autors [5].

Ein interaktives graphisches Tool zum Zusammensetzen derartiger Karten ist das vom Autor entwickelte Programm *QBigMap* [6]. Zusätzlich können hier auch GPX-Tracks dargestellt und -Routen entworfen werden.

Ein Nachteil der auf diese Weise generierten Karten soll nicht verschwiegen werden: Da die Kacheln für die Online-Darstellung optimiert sind, erscheinen Beschriftungen im Ausdruck häufig sehr klein oder nur schwer lesbar.

Eigene Online-Karten

Für Weg- und Ortsbeschreibungen auf Webseiten besteht häufig der Wunsch, eigene Online-Karten bereitzustellen. Dazu ist etwas *JavaScript*-Programmierung erforderlich. Die Kartenfunktionalität wird hier von den Bibliotheken *OpenLayers* [7] und *Leaflet* [8] bereitgestellt, wobei die letztere sehr einfach zu nutzen, die erste allerdings leistungsfähiger ist.

Für die Bereitstellung der Karten genügt ein einfacher Webspace, es wird keinerlei zusätzliche Serverfunktionalität benötigt.

Eine eigene Karte unter Benutzung der *Leaflet*-Bibliothek besteht aus einem HTML- und einem Javascript-Teil, der hier auf zwei Dateien verteilt wird.

Die HTML-Datei bindet die benötigten JavaScript-Bibliotheken und Styles ein und stellt die Kartenfläche in Form eines `div`-Elements mit vorgegebener Größe bereit.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Leaflet-Karte 1</title>
<!-- Leaflet-Symbole -->
<link rel="stylesheet"
      href="http://cdn.leafletjs.com/leaflet-0.4/leaflet.css"/>
<!-- Leaflet-Bibliothek -->
<script src="http://cdn.leafletjs.com/leaflet-0.4/leaflet.js"></script>
<!-- Eigene Kartenfunktionalität -->
<script src="myleaflet1.js"></script>
</head>
<!-- Initialisierung mit Breite, Länge, Zoom -->
<body onload="javascript:map(50.8,12.9,12)">
<!-- Kartenfläche, Größe muss gesetzt sein -->
<div id="map" style="width:100%;height:100%;"></div>
</body>
```

Die JavaScript-Datei erzeugt das Kartenobjekt und verknüpft es mit einer oder mehreren Kartenquellen:

```
function map(lat, lon, zoom) {
  // Kartenobjekt mit Position und Zoomfaktor erzeugen
  var myMap = L.map('map').setView([lat, lon], zoom);
  // Attributionstring (Copyright) setzen
  var attribution = 'Map␣data␣&copy;␣<a␣href="http://openstreetmap.org">'
    +'OpenStreetMap</a>␣contributors,␣'
    +'<a␣href="http://opendatacommons.org/licenses/odbl/">ODbL</a>';
  // Kartenlayer hinzufügen, Platzhalter {z}, {x}, {y}
  L.tileLayer('http://tile.openstreetmap.org/{z}/{x}/{y}.png',
```

```
  { attribution: attribution,
    maxZoom: 18
  }).addTo(myMap);
}
```

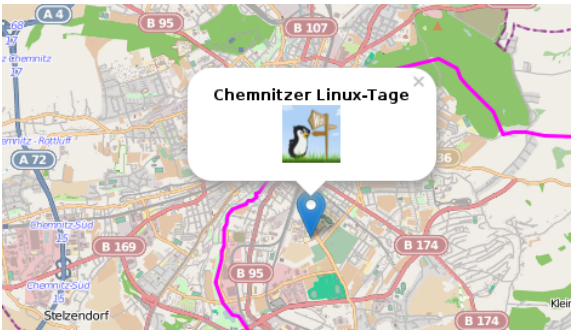
Um einen Punkt (*POI* = Point of Interest) zu markieren, wird folgender Code ergänzt:

```
// Setze Marker
var marker = L.marker([50.8135, 12.9293]).addTo(myMap);
// Beim Anklicken Popup anzeigen
marker.bindPopup('<b>Chemnitzer Linux-Tage<br/><img src="clt-logo.png"/>');
```

Um einen Track anzuzeigen, muss dieser mittels `XMLHttpRequest`-Objekt geladen und anschließend geparst werden.

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.open('GET', 'track.gpx', true);       // URL des Tracks
xmlhttp.overrideMimeType('application/xml'); // Rückgabetyp setzen
xmlhttp.onreadystatechange = function () {
  if (xmlhttp.readyState != 4) return;        // Request unvollständig
  if (xmlhttp.status != 0 && xmlhttp.status != 200 && xmlhttp.status != 304)
    return; // Fehler
  // Request erfolgreich, GPX-Punkte ermitteln.
  // Für Waypoints "wpt", für Routenpunkte "rtept" statt "trkpt" verwenden.
  var nodes = xmlhttp.responseXML.getElementsByTagName('trkpt');
  drawNodes(map, nodes);
}
xmlhttp.send();

function drawNodes(map, nodes) {
  poly = [];    // Koordinatenfeld
  for (var i = 0; i < nodes.length; i++) {
    var lat = parseFloat(nodes[i].getAttribute('lat'));
    var lon = parseFloat(nodes[i].getAttribute('lon'));
    poly.push(new L.LatLng(lat, lon)); // Koordinate hinzufügen
  }
  var track = new L.Polyline(poly, {}).addTo(map);  // Linienzug hinzufügen
  track.bindPopup('Trackbeschreibung');             // Popup für Linienzug
}
```



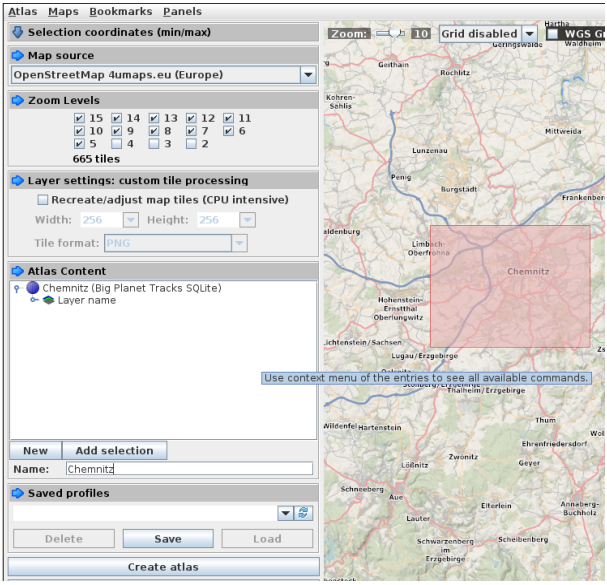
Online-Karte mit Leaflet, Popup und Track

Android-Karten

Für Android-Geräte existieren eine Vielzahl freier und kommerzieller Apps auf der Basis von OpenStreetMap-Daten. Hier wurde die quelloffene, kachelbasierte Anwendung *Big Planet Tracks* [9] ausgewählt, die Offline-Nutzung und das Aufzeichnen von Tracks ermöglicht.

Die Atlanten werden in Form von SQLite3-Datenbanken bereitgestellt, die Auswahl zwischen mehreren Atlanten ist möglich. Beim Online-Betrieb werden diese Datenbanken als lokaler Cache benutzt, im Offline-Betrieb dienen sie als Datenquelle.

Offline-Karten können auf dem PC mit der Java-Anwendung *Mobile Atlas Creator* (MOBAC) [10] für gewünschte Regionen und Zoomstufen erzeugt werden.



MOBAC-Kartenerzeugung

Nach dem Start dieser Java-Anwendung mit dem Befehl

```
java -jar Mobile_Atlas_Creator.jar
```

werden ein Atlasname und das Ausgabeformat *Big Planet Tracks SQLite* gewählt. Sind Kartenquelle, die gewünschten Zoomstufen und die Region gewählt, kann der Atlas erzeugt werden, was je nach Zahl der benötigten Kacheln etwas Geduld erfordert. Die Karte wird im Verzeichnis `~/atlases` abgelegt und muss nun in das Massenspeicherverzeichnis `RMaps/maps` übertragen werden, was über die USB-Verbindung erfolgen kann. Wenn das Gerät für die Datenübertragung nur den MTP-Modus unterstüzt, muss unter Debian das Paket `mtpfs` installiert und anschließend das Gerät manuell montiert werden:

```
sudo mtpfs -o allow_other /mnt
```

Nach dem Kopieren der Datei wird das Gerät wieder ausgehängt:

```
sudo umount /mnt
```

MOBAC stellt jedoch nur bestimmte Kartenstile zur Auswahl, eine Konfigurationsmöglichkeit für weitere Datenquellen konnte nicht gefunden werden. Deshalb soll das Datenformat der SQLite-Datenbank kurz erläutert werden, um erfahrenen Anwendern die Bereitstellung eigener Atlanten aus anderen Kachelquellen und mit Overlays zu ermöglichen.

Die Datenbank besteht aus drei Tabellen. Die Tabelle `android_metadata` enthält lediglich die Spalte `locale` mit einem einzigen Eintrag der Sprache (`de_DE`), die Tabelle `info` die beiden Spalten `minzoom` und `maxzoom`, die den kleinsten und größten Zoomfaktor der gespeicherten Daten enthalten. Die eigentlichen Daten befinden sich in der Tabelle `tiles`. Hier enthalten die Spalten `x`, `y` und `z` die Kachelnummer der jeweiligen Zoomstufe und die Spalte `image` die Kacheldaten als JPEG- oder PNG-Bild. Die Spalte `s` (Strategy) enthält bei Offline-Karten immer den Wert `0`.

Karten für Garmin-GPS-Empfänger

Viele Garmin-GPS-Empfänger ermöglichen das Laden eigener Karten, die nach Vorstellung des Herstellers von diesem gekauft werden sollen. Inzwischen wurde jedoch das Datenformat weitgehend entschlüsselt, so dass eigene Karten erstellt werden können. Im *OpenStreetMap*-Wiki [1] stehen für eine Vielzahl von Regionen solche Karten zum Download zur Verfügung. Ist eine Karte für eine gewünschte Region jedoch nicht verfügbar oder aufgrund der Größe für ein etwas älteres Gerät nicht geeignet, so kann diese selbst berechnet werden. Voraussetzung für die Nutzung der Werkzeuge ist eine installierte Java-Runtime-Umgebung.

Im Gegensatz zu den bisher beschriebenen Anwendungen benötigen *Garmin*-Geräte vektorbasierte Karten, die aus den *OpenStreetMap*-Vektordaten erzeugt werden. Diese werden einmal pro Woche für den ganzen Planeten und täglich für einzelne Kontinente, Länder und Regionen bereitgestellt [11]. Aufgrund der Datenmenge sollte man sich auf ein möglichst kleines Gebiet beschränken, das die Region der benötigten Garmin-Karte umfasst, und das `PBF`-Dateiformat verwenden.

Aus diesem Gebiet wird mithilfe des Java-Werkzeugs *Osmosis* die Kartenregion herausgeschnitten.

```
osmosis --rb file="region.osm.pbf" \
  --bb left=x0 bottom=y0 right=x1 top=y1 clipIncompleteEntities=true \
  --wx file="karte.osm"
```

Da diese Operation sehr lange dauert, bietet sich für kleinere Regionen (Urlaubsgebiete, Bundesländer) alternativ eine Online-Abfrage mithilfe der Overpass-API [12] an.

```
curl --data-urlencode "data=\
( node(süd,west,nord,ost);\
  <;\
);\
out;" \
-o karte.osm http://overpass-api.de/api/interpreter
```

Diese muss mit dem Werkzeug *Splitter* weiter zunächst in Teilregionen zerlegt werden.

```
java -Xmx2048M -jar splitter.jar karte.osm
```

Gemeinsam mit den Datendateien wird eine Steuerdatei `template.args` erzeugt, die für das Generieren der Karte benötigt wird.

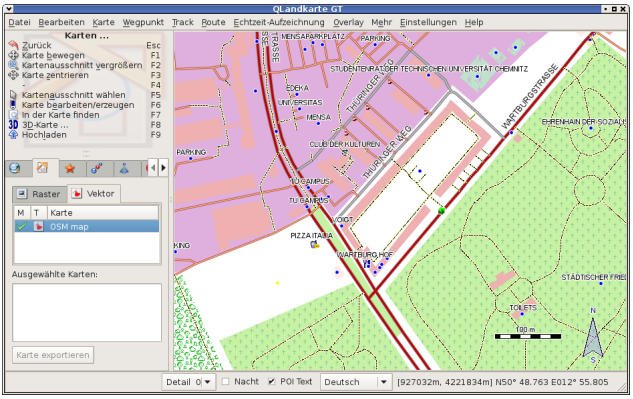
Für die eigentliche Kartenerzeugung werden noch einen Kartenstil und eine zugehörige `TYP`-Datei, die das Kartenlayout definieren, benötigt. Da eine eigene Erstellung nicht ganz einfach ist, bietet es sich an, auf vorhande Vorlagen zurückzugreifen, hier werden die des OpenStreetMap-Nutzers *Computerteddy* [13] genutzt. Die Dateien werden im gleichen Verzeichnis wie die Kartendaten abgelegt bzw. ausgepackt.

Der Aufruf

```
java -Xmx2048M -jar /opt/mkgmap/mkgmap.jar \
  --max-jobs \
  --remove-short-arcs \
  --latin1 \
  --reduce-point-density=10 \
  --mapname="Kartenname" \
  --gmapsupp -c template.args \
  --family-id=42 --style-file=teddy teddy.typ
```

erzeugt eine Datei `gmapsupp.img` mit der fertigen Karte.

Ist die Karte generiert, kann sie mit dem Programm *QLandkarteGT* kontrolliert werden. Dazu muss zunächst die Datei `osmmap.tdb` und anschließend die eigentliche Karte `gmapsupp.img` geladen werden.



Garmin-Karte in QLandkarteGT

Diese fertige Karte wird nun im Massenspeichermodus auf das Garmin-Gerät übertragen und im Verzeichnis `garmin` der SD-Karte abgelegt. Nach einem Reboot des Gerätes kann die neue Karte genutzt werden.

Literatur und Online-Ressourcen

[1] http://wiki.openstreetmap.de – OpenStreetMap Wiki.

[2] Ramm, Topf: *OpenStreetMap. Die freie Weltkarte nutzen und gestalten.* 3. Auflage 2010. Lehmanns New Media.

[3] Robinson et al: *Elements of Cartography.* 6th ed. 1995. John Wiley & Sons.

[4] https://addons.mozilla.org/de/firefox/addon/live-http-headers/ – Live HTTP Headers

[5] http://ruessel.in-chemnitz.de/osm/bigmap.html – Python-Script `bigmap.py`.

[6] http://ruessel.in-chemnitz.de/osm/qbigmap – QBigMap.

[7] http://www.openlayers.org – JavaScript-Bibliothek OpenLayers.

[8] http://leafletjs.com – JavaScript-Bibliothek Leaflet.

[9] http://code.google.com/p/big-planet-tracks/ – Android-App Big Planet Tracks.

[10] http://mobac.sourceforge.net/ – Mobile Atlas Creator.

[11] http://download.geofabrik.de/ – Download von OpenStreetMap-Vektordaten.

[12] http://overpass-api.de – Overpass API.

[13] http://wiki.openstreetmap.org/wiki/User:Computerteddy – Computerteddys Typ- und Styledateien für Garmin-Geräte.

[14] http://www.mkgmap.org.uk/ – mkgmap.

[15] http://ruessel.in-chemnitz.de/osm/clt2013 – Webseite zum Vortrag.