TK104 PLUGIN FÜR OWNTRACKS RECORDER

2. NOVEMBER 2016

BJÖRN PRÖPPER

Inhalt

Einleitung

Der Personentracker nutzt das Duo Mosquitto[1] und ot-recorder [2]als System zur Erfassung und zur Darstellung von Positionsdaten. Um die Daten des GSM/SMS basierten Tracker TK104 verarbeiten und darstellen zu können ist ein Plug-In notwendig, dass zum einen die Daten empfangen kann (SMS), in das MQTT Format überträgt und dann an den Server sendet.

Um das System zu implementieren wurde ein SMS-Server auf einem Raspberry Pi 3 aufgesetzt, der die SMS des Trackers mit den Positionsdaten empfängt, umwandelt und dann an das ot-recorder Backend versendet. Es besteht die Möglichkeit das Plug-In auf demselben Rechner laufen zu lassen wie das Backend, oder auf einem extra Rechner, der dann aber den Server über das Internet erreichen können muss.

Systembeschreibung

Der Tracker TK104 ist dazu in der Lage auf verschiedenen Wegen seine Position zu versenden. Zum einen kann eine Positionsabfrage durch einen Anruf bei dem Tracker ausgelöst werden, dann kann über eine SMS eine einzelne Positionsmeldung abgerufen werden, oder der TK104 kann so per SMS konfiguriert werden, dass er automatisch in definierten Zeitabständen seine Position sendet. Es besteht die Möglichkeit die Daten auch per GPRS zu versenden, diese Option ist in diesem Pulg-In aber noch nicht implementiert und Bedarf der weiteren Entwicklung.

Der TK104 nutzt zum Übertragen der Daten das GSM Netz. Er verfügt über eine SIM-Karte und benötigt dementsprechend ein Mobiltelefon als Gegenstelle. Dazu kann man sowohl ein Mobiltelefon verwenden und dies über USB an den Raspberry Pi SMS anschließen, oder einen USB-Datenstick. Die Versuche mit dem USB Datenstick waren allerdings nicht besonders erfolgreich und bedürfen der weiteren Entwicklung.

Die meisten Mobiltelefone verstehen den AT- Befehlssatz[3] und sind damit kompatibel. Auch die Satellitentelefone von Thuraya[4] verstehen den AT-Modembefehlssatz und sollten damit auch kompatibel sein.

In diesem Dokument wird zunächst nur die Option beschrieben die Daten per SMS vom TK104 zu empfangen und zu verarbeiten. Dazu ist es notwendig zunächst die Möglichkeit zu schaffen SMS auf dem Raspberry zu empfangen, diese nach Empfang zu verarbeiten und dann an das ot-recorder Backend zu senden.

Für den Empfang der SMS wird die Software „gammu-smsd[5]“ verwendet. Diese ist dazu in der Lage SMS über ein angeschlossenes USB-Gerät zu versenden und zu empfangen. Beim Empfang einer SMS kann automatisch ein Skript gestartet werden, dass die empfangenen Daten verarbeitet. Von dieser Funktion wird hier Gebrauch gemacht.

---

[1] https://mosquitto.org/
[2] http://owntracks.org/booklet/guide/apps/ Download unter: https://github.com/owntracks/recorder oder https://itunes.apple.com/de/app/owntracks/id692424691?mt=8
[3] https://de.wikipedia.org/wiki/AT-Befehlssatz
[4] http://www.thuraya.de/
[5] https://wammu.eu/smsd/

Die Umsetzung der Daten und auch das Versenden an das Backend erfolgt über ein selbstgeschriebenes Skript. Das Skript erwartet die zu verarbeitenden Daten als Datei, deren Name als Argument an das Skript übergeben wird. Der TK104 nutzt vier verschiedene Formate, um die Daten zu versenden. Deshalb ist eine Fallunterscheidung notwendig, bevor die Daten in ein für das Backend verständliches Format umgesetzt werden können.

Die verschiedenen Formate sehen wie folgt aus:

1. Tracker wird angerufen und kann eine GPS ermittelte Koordinate zurücksenden:

```
lat:34.527582 lon:69.175632
speed:0.00
T:16/10/30 16:36
bat:100%
http://maps.google.com/maps?f=q&q=34.527582,69.175632&z=16
```

2. Tracker wird angerufen und kann KEINE GPS ermittelte Koordinate zurücksenden:

```
Last:
lat:34.527397
lon:69.175705
T:17:45
http://maps.google.com/maps?f=q&q=34.527397,69.175705&z=16
Now:
Lac:277d 2898
T:16/10/30 16:13
bat:100%
```

3. Tracker wird per SMS angefragt und kann eine GPS ermittelte Koordinate zurückliefern:

```
lat:34.527583
long:69.175635
speed:0.00
T:16/10/30 16:40
bat:100%
http://maps.google.com/maps?f=q&q=34.527583,69.175635&z=16
```

4. Tracker wird per SMS abgefragt und kann KEINE GPS ermittelte Koordinate zurückliefern:

```
Lac:277d 2898
T:16/11/05 09:28
Last:
T:17:22
http://maps.google.com/maps?f=q&q=34.527608,69.175577&z=16
```

Zu Debugging-Zwecken werden die Daten in einer Datei abgelegt, die den Namen des Gerätes, identifiziert durch die Telefonnummer enthält.

Nach Umwandlung der Daten wird durch das Skript eine Verbindung zu dem Server aufgebaut und die Daten werden versendet. Dabei wird die Telefonnummer des Tracker als Gerät im Owntracks Topic verwendet. Da die Verbindung IP-basiert ist, spielt es keine Rolle ob das Backend auf dem gleichen Rechner läuft (localhost), oder über das Internet erreichbar ist.

Die untenstehende Abbildung zeigt den grundsätzlichen Aufbau der Lösung:

Von links nach rechts sind der TK104, das Mobiltelefon als Gegenstelle, der Raspberry Pi und dessen Stromversorgung zu erkennen.

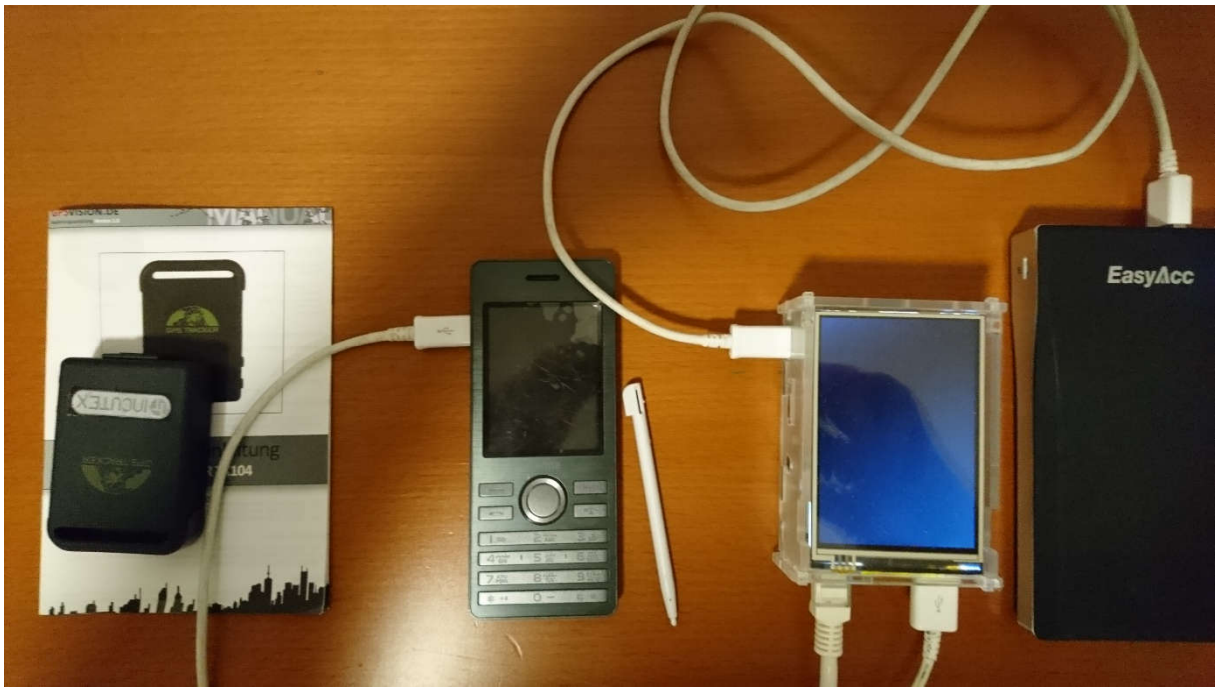

Abbildung 1 Das Setup des Plug-Ins

Voraussetzungen

Ein Server auf dem ein MQTT Broker auf Port 8883 aktiv ist.

Der Server muss über das Internet erreichbar sein, d.h. man muss ggf. eine Portweiterleitung im Router einrichten.

Das Plug-In ist in Python 2.7 geschrieben. In Python muss die Bibliothek Paho[6] installiert sein.

Die anderen verwendeten Bibliotheken scheinen zum normalen Umfang einer Python-Dokumentation zu gehören und sind daher automatisch verfügbar.

Installation

Auf dem Raspberry Pi wurde ein aktuelles auf Debian „Jessie“ basiertes Raspian Image installiert.

Das Telefon, in diesem Fall ein Ashna MTK2 wird über einen USB Port angeschlossen. Die Internetanbindung erfolgt beim vorliegenden Fall über den Ethernet-Anschluss des Raspberry.

Zunächst ist die Software gammu-smsd zu installieren:

```
sudo apt install gammu-smsd
```

Danach ist die Konfigurationsdatei zu erstellen, bzw. die Standardkonfiguration in /etc/gammusmsdrc anzupassen. Die verwendete Konfiguration ist unter Anlagen/Konfiguration Gammu-smsd zu finden.

[6] https://pypi.python.org/pypi/paho-mqtt/1.1

Das Skript zum Verarbeiten der Daten wird dabei über den Parameter RunOnReceive angegeben. Es muss sich um ein ausführbares Skript handeln, dass auch für den Nutzer „gammu“ verfügbar sein muss. Die Konfiguration für die Übergabe an den Server befinden sich in der Datei „receiver_config.py“

Danach ist die Software als Dienst einzurichten

```
sudo systemctl enable gammu-smsd
```

und zu starten

```
sudo systemctl start gammu-smsd
```

Nutzung des TK104

Der TK104 kann über den Raspberry konfiguriert und abgefragt werden. Die dazu notwendigen Kommandos sind in der Dokumentation des TK104 dargestellt. Um die entsprechenden Kommandos an den TK104 zu versenden ist die Software gammu-smsd-inject zu verwenden, die mit dem Paket gammu-smsd installiert wird.

Der Aufruf erfolgt dabei wie folgt:

```
echo "Kommando" | sudo gammu-smsd-inject TEXT "+Telefonnummer"
```

Beispiel:

```
echo "fix001m001n123456" | sudo gammu-smsd-inject TEXT "XXXXXXXXXXX"
```

Sendet eine SMS an den TK104, mit dem einmal eine Position abgerufen wird.

Anlagen

Konfiguration Gammu-smsd

```
# Configuration file for Gammu SMS Daemon
# Gammu library configuration, see gammurc(5)
[gammu]
# Please configure this!
port = /dev/ttyUSB0
connection = at
CheckSecurity=0
# SMSD configuration, see gammu-smsdrc(5)
[smsd]
CheckSecurity=0
service = files

logfile = /var/log/gammu-smsd
# Increase for debugging information
debuglevel = 4
ReceiveFrequency = 30
# Paths where messages are stored
inboxpath = /var/spool/gammu/inbox/
outboxpath = /var/spool/gammu/outbox/
sentsmspath = /var/spool/gammu/sent/
errorsmspath = /var/spool/gammu/error/
RunOnReceive = /home/pi/receive.py

CheckBattery = 0
CheckSignal= 0
CheckNetwork = 0
```

Skriptcode

Der Skriptcode stellt die Version 0.1 des Skriptes receive.py dar, die jeweils aktuellste Version befindet sich auf GitHub unter: https://github.com/btesteracc/Tk104

```
#!/usr/bin/env python
from __future__ import print_function
import os
import sys
import json
import datetime
import paho.mqtt.publish as mqtt
import re

import receiver_config as conf

data={
    "_type" : "location",
    "acc"   : 0,
    "alt"   : 0,
    "batt"  : 0,
    "cog"   : 0,
    "desc"  : "",
    "event" : "",
    "lat"   : 0.0,
    "lon"   : 0.0,
    "rad"   : 0,
    "t"     : "x",
```

```
    "tid"   : "bp",
    "tst"   : 0,
    "vacc"  : 0,
    "vel"   : 0,
    "p"     : 0
}

InFilename="/var/spool/gammu/inbox/"+sys.argv[1]
with open(InFilename,'r') as InFile:
    content=InFile.readlines()
InFile.close()
Devicename=sys.argv[1].split("_")[3]
conf.topic+=Devicename[1:]
if content[0].startswith("Last"):
    data["lat"]=float(content[1][4:])
    data["lon"]=float(content[2][4:])
    data["batt"]=int(content[-1][4:-2])
    data["desc"]=content[6]+content[3]
    dateline=-2
    #checked i.O.

elif (content[0].startswith("lat")) and (content[0].find("lon")!=-1):
    grids=re.findall("([-+]?\d{1,2}[.]\d+)",content[0])
    data["lat"]=float(grids[0])
    data["lon"]=float(grids[1])
    data["batt"]=int(content[3][4:-2])
    dateline=2
    #checked i.O.

elif content[0].startswith("lat") and content[1].startswith("long"):
    data["lat"]=float(content[0][4:])
    data["lon"]=float(content[1][5:])
    data["batt"]=int(content[4][4:-2])
    dateline=3
    #checked i.O.

elif content[0].startswith("Lac"):
    dateline=1
    grids=re.findall("([-+]?\d{1,2}[.]\d+)\,([-+]?\d{1,2}[.]\d+)",content[4])
    data["lat"]=float(grids[0][0])
    data["lon"]=float(grids[0][1])
    data["desc"]=content[0]+content[4]
    #checked i.O.

format="T:%y/%m/%d %H:%M\n"
date_obj=datetime.datetime.strptime(content[dateline],format)
data['tst'] = date_obj.strftime('%s')
if (data['tst']!=0) and (data['lat']!=0.0) and (data['lon']!=0.0):
    datastr=json.dumps(data)
    #Zum Testen auskommentiert am 01.11.2016
    # Ziel ist es mal den ganzen Tag Daten zum empfangen und diese nur lokal
abzulegen


#mqtt.single(conf.topic,payload=datastr,auth=conf.userid,port=conf.port,hostna
me=conf.hostname)
```

```
    #aus diesem Grund wird die Datei auch unter var/spool/gammu abglegt und
nicht mehr im tmp Verzeichnis
    #with open("/tmp/receive.txt",'a') as OutFile:
    with open("/var/spool/gammu/"+Devicename[1:],'a') as OutFile:
        OutFile.write(datastr+'\n')
    OutFile.close()
```

Die Konfiguration für den Zugriff auf den Server sieht wie folgt aus:

```
userid ={'username':"bjoern",'password':"123456"}
topic="owntracks/topic/"
hostname="localhost"
port=8883
```

Abbildungsverzeichnis:

Änderungsnachweis

Datum/Version	Änderung	Verfasser
2.11.2016 V1.0	Erstellung	Björn Pröpper
05.11.2016 V1.1	Ergänzung der Fallunterscheidungen	Björn Pröpper