Per LoRaWAN im Internet der Dinge über weite Distanzen kommunizieren

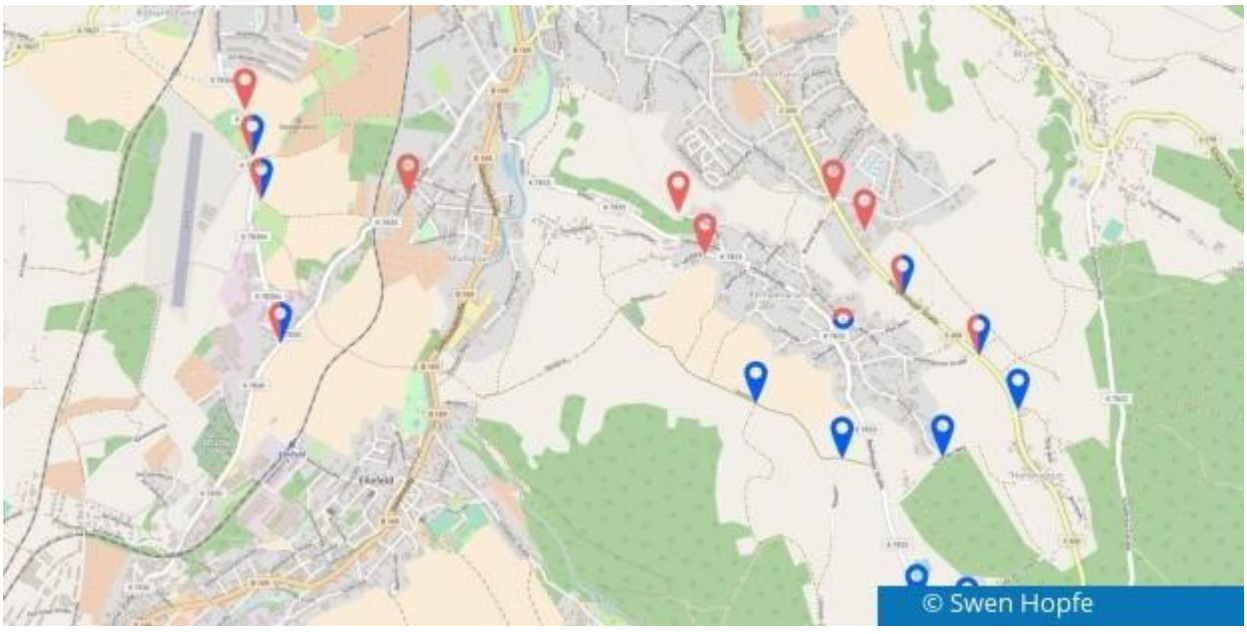
14.07.2018 | 09:44 Uhr | Swen Hopfe ()

Swen Hopfe

Swen Hopfe arbeitet bei einem mittelständischen Unternehmen aus der Chipkarten-Branche und ist Experte für Smart Cards, RFID, das IoT, Raspberry Pi und Arduino. Mehr (/autoren/swen_hopfe_1523011113498823183)

Autorenprofil schließen ()

Wer im Internet der Dinge über weite Entfernungen Daten senden und empfangen will, kann dazu das LoRaWAN nutzen. Wie gut das klappt, hat unser Autor getestet.



LoRa-Technologien im Wide-Area-Network (https://www.lora-wan.de/) werden in diesen Tagen immer interessanter, um im „Internet der Dinge“ verschiedene Geräte auf längeren Distanzen miteinander kommunizieren zu lassen. Betrachtet man nur die Reichweite, so sind sie zwischen dem lokalen Netzwerk und der terrestrischen Video- oder Audioübertragung (Radio, DVB-T, Funk von Notdiensten etc.) angesiedelt.

Die Übertragung geschieht digital. LoRaWAN meint dabei gleichzeitig das Netzwerkprotokoll, das auf Low-Power-Technologien ausgerichtet wurde. Die Spezifikation dazu wurde von der LoRa-Alliance

(https://www.lora-alliance.org/) aufgelegt und ist frei verfügbar. Hardware, die solche Modulationstechnik nutzt, kann typischerweise Verbindungen von 2 Kilometer bis 40 Kilometer herstellen.

Warum diese Technik also nicht im Selbstversuch einmal ausprobieren, mit dem Ziel, über eine möglichst weite Entfernung eine Verbindung zu bewerkstelligen?

Zur Ortsbestimmung zwecks genauer Entfernungsmessung soll gleich GPS zum Einsatz kommen. Später soll überlegt werden, wie die Datenübertragung für weitere Zwecke genutzt werden kann...

Eigener Aufbau

LoRaWAN nutzt regional unterschiedliche Frequenzbereiche im ISM-Band. In Europa sind die Frequenzen 433 MHz (ISM-Band Region 1) und 868 MHz (SRD-Band Europa) für die Datenübermittlung freigegeben.

Unser Sender baut auf der interessanten Feather-Reihe von Adafruit (https://www.adafruit.com/feather) auf, die man wie einen Arduino programmieren kann. Das Platinenformat in doppelter Briefmarkengröße orientiert sich dann auch an einem Arduino-Nano.

Wir haben die Cortex-M0-Variante gewählt, die es unter anderem gleich in Kombination mit einem RFM95-LoRa-Chipsatz mit 868 MHz gibt. Daneben sind mittlerweile viele Feathers und Featherwings erhältlich - also steckbare Zusatzmodule. Für die ersten Gehversuche war für uns ein GPS-Modul das Wichtigste, denn unser Sender soll ja seinen Standort übermitteln.

Alle Feather mit LoRa-Chipsatz sind als Transceiver aufgebaut, können also senden und empfangen. Das macht die Entwicklung eines Pärchens oder Netzwerk aus Hardware-Sicht um einiges einfacher.

Damit das Ganze senderseitig mobil wird, lässt sich günstigerweise noch ein LiPo-Akku standardmäßig anschließen. Soll geladen werden, steckt man einfach das Netzteil am Micro-USB-Port des Feather an, und die integrierte Ladeelektronik informiert dann per LED über den aktuellen Zustand.

Möchte man mehrere Module übereinander schichten, sollte man sich vorher über Stift- und Buchsenleisten Gedanken machen, damit das dann auch funktioniert. Eine Antenne ist anzulöten. Die Drahtvariante reicht uns für den Anfang, der Sender soll ja auch kompakt bleiben.

Für den Entwurf und das Aufspielen der Firmware reicht die Arduino-IDE. Aber für das Projekt hier [illegible] ich von Anfang an eine komfortablere Lösung. Die schafft man sich unter Linux durch den Atom-Editor und das PlatformIO-Plugin.

Hat man alle Packages für den Cortex M0 installiert, ist das Ganze wirklich komfortabel, denn man kann mehrere Projekte, wie in unserem Fall das Sender- und Empfänger-Projekt, zentral verwalten und für jedes getrennte Terminals aufmachen. Alle Entwickler, die „Sublime“ als Werkzeug nutzen, werden die augenschonende Oberfläche wiedererkennen.

Die Firmware für unseren Sender ist dann auch relativ klein, hat aber ein paar besondere Details eingebaut, die wir schon früher im Umgang mit GPS-Modulen erprobt haben.

Zuerst wird auf einen GPS-Fix gewartet, dann eine Anfrage an den Empfänger gestartet. Der sendet ein entsprechendes Codewort. Ist der Challenge-Response-Test erfolgreich gewesen, werden die empfangenen GPS-Daten in XML gehüllt und schrittweise übertragen. Logisch, dass man sich an dieser Stelle jeweils eigene Gedanken über die Sicherheit beim Handshaking und über die Übertragungssicherheit machen muss.

Nimmt der Sender Fahrt auf, werden bei uns die Sendeabstände korrigiert. Bei langsamer Bewegung im Fußmarsch werden nur alle 10 Sekunden Daten übermittelt, ab 25 km/h schon alle Sekunde. Differiert der ermittelte Standort nur über ein paar Meter, wird gar nichts per Funk abgesetzt.

Auch wenn wir in unserer Karte in der Zusammenfassung keine Tracks aufgezeichnet haben (sondern nur ausgewählte Punkte), kann man bei ständiger Aufzeichnung damit dynamisch anpassen, den Traffic an der Quelle begrenzen und gegen diverse Ungenauigkeiten bei der Lagebestimmung ansteuern, was sich in der Praxis gut bewährt hat.

Wie sieht es nun mit dem Empfänger aus? Auch der hat bei uns einen Feather als Grundlage und eine etwas größere Empfangsantenne spendiert bekommen. Daneben noch ein OLED-Display.

Das OLED ist ein Wing-Modul und gibt diverse Statusmeldungen aus. Es nutzt den I2C-Port des Feather. Man muss es nicht unbedingt stapeln, sondern kann es auch frei verdrahten, für den Anfang auf einem Steckboard zum Beispiel. Von dem, was vom Sender ankommt, wird nur ein Teil auf dem Display angezeigt, etwa die neuen Koordinaten und der RSSI-Faktor, damit man beim Einrichten die Stärke der Funkverbindung einschätzen kann.

Alles Restliche leitet unser Empfänger auf seine serielle Schnittstelle weiter. Das ist bei uns der USB-Port, gleich dem Stromanschluss an der Micro-USB-Buchse. Zu Testzwecken kann man den an einen PC anschließen und auf einem Terminalprogramm mithören.

Wir haben „moserial“ auf einem Notebook mit Linux-Mint genutzt, weil es eine einfache Möglichkeit bietet, alles Empfangene in eine Datei zu schreiben. Unter Windows (/handover/451?ws=1) geht das genauso.

Während unser Sender mit Akku schon autark ist, wollten wir die Empfangsseite nun auch von einem unhandlichen Rechner entkoppeln. Da bot sich ein Pi an, der nun die empfangenen Daten vom Feather managen soll, die er per USB (/handover/601?ws=1) anstatt des Notebook bekommt. Da unser Hausserver auch ein Raspberry Pi 3 ist, der schon 24/7 läuft, stellte das einen interessanten Testfall für mich dar, um für die Zukunft einen ständigen LoRa-Knotenpunkt zu Hause aufzubauen.

Feather am USB-Port des Pi

Nach dem Stecken des Empfangs-Feather an den Pi sollte eine Ausgabe von **dmesg** einen solchen oder ähnlichen Part enthalten:

```
[4.080600] usb 1-1.3: new full-speed USB device number 4 using dwc_otg
[4.221811] usb 1-1.3: New USB device found, idVendor=239a, idProduct=800b
[4.221821] usb 1-1.3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[4.221828] usb 1-1.3: Product: Adafruit Feather M0
[4.221835] usb 1-1.3: Manufacturer: Adafruit
[4.331209] cdc_acm 1-1.3:1.0: ttyACM0: USB ACM device
[4.334197] usbcore: registered new interface driver cdc_acm
```

Um auf dem Pi von USB (/handover/601?ws=1) zu lesen, kann man **minicom** nutzen, man installiert das Terminalprogramm mittels

```
$ sudo apt-get install minicom
```

Nun kann **minicom** auf dem Port des Raspberry Pi, wo unser Feather angeschlossen ist, geöffnet werden:

```
$ sudo minicom -b 115200 -D /dev/ttyACM0 -o
```

Mit einem kleinen Python-Script geht es auch:

```
#!/usr/bin/env python
import serial
import tty
import sys
import time
def main():
    obj_out = open("trip.gpx","w")
```

```
    ser = serial.Serial('/dev/ttyACM0', 115200)
    while True:
        line = ser.readline()
        print line
        fobj_out.write(line)
        time.sleep(1)
if __name__ == '__main__':
    main()
```

Dazu muss vorher noch python-serial installiert werden:

```
$ sudo apt-get install python-serial
```

Dann funktioniert unser Code, wenn man ihn beispielsweise in einer Datei **lora_rec.py** untergebracht hat:

```
$ python lora_rec.py
```

Das provisorische Script kann man mit <Ctrl-C> abbrechen, beim Schreiben der Datei gibt es meist keine Probleme. Für Testzwecke brauchen wir nicht mehr. Da der Sender bereits XML übermittelt, muss für eine reguläre Datei nur noch

</trkseg>

</trk>

</gpx>

an die eben geschriebene **trip.gpx** angehängt werden. Und schon kann man das Ganze in einem GPX-Viewer anschauen.

Ergebnisse

Da wir unseren Empfänger im Haus stationiert haben, hier nun eine Landkarte mit blauen und roten Markierungen. Die beiden verschiedenen Farben zeigen an, welche entfernten Sender-Standorte empfangbar waren, je nachdem, an welcher Gebäudeseite der Empfänger platziert war. Nicht unerheblich bei uns, da wir im Tal liegen und nur die zwei Giebelseiten für den Empfang richtig geeignet sind. Den Empfänger-Standort gibt der farbige Ring an.

Durch das Python-Script auf dem Pi sind ja komplette Tracks geloggt worden, die Karte hier ist also nur zur Illustration der interessanten Punkte.

Wenn man bedenkt, dass der Sender mit einer einfachen Drahtantenne und im PKW gearbeitet hat und auch der Empfänger hinter Glas stand, finde ich das Ergebnis absolut zufriedenstellend. Der Theorie kommt man wie immer nur nahe, wenn man optimale Bedingungen schaffen kann.

Letztendlich konnte eine Verbindung über knapp 4 Kilometer aufgebaut werden. In der Ebene und im Freiland erzielt man sicher bessere Ergebnisse.

Ausblick

In diversen Aufbauten ist schon über richtig große Distanzen hinweg kommuniziert worden. Einiges davon findet man im Web. Auch über diverse Gateways, in denen LoRaWAN heutzutage schon betrieben wird.

Fertige Gateways mit TTN-Kompatibilität sind zum Beispiel „Lorank 8“, „Lorix One“ oder „The Things Gateway“. Diese zeichnen sich durch eine sternförmige Netzarchitektur aus. Die jeweiligen Endgeräte

kommunizieren mit dem Gateway, welches die Datenpakete an einen Server sendet. Dieser ist dann über Schnittstellen mit diversen IoT-Anwendungen verbunden.

Recht teure Geräte dafür kann man sich auch kaufen. Dazu muss man aber auch in Reichweite eines solchen Netzwerks sein, was bei uns nicht der Fall ist.

Daneben kann LoRa aber auch für eine Punkt-zu-Punkt-Verbindung dienen oder unabhängige lokale Netzwerke spannen wie in Umweltschutz oder -beobachtung, zum Beispiel im Forst. Sicherlich auch interessant ist es für die Landwirtschaft und den Personennahverkehr, außerdem können verteilte Wohnanlagen oder dezentrale Energieerzeuger kostengünstig miteinander verbunden werden. Im Privaten ist es vielleicht die Verbindung zum Schrebergarten vom Wohnhaus aus oder der Diebstahlschutz.

Neben GSM oder WiFi warten auf mittlere Entfernungen also eine Vielzahl von Anwendungsmöglichkeiten auf die Realisierung, wie in unserem "Proof of Concept" hier gezeigt. Und der soll für mich auf jeden Fall Ansporn für neue Projekte sein...