Ein einfaches Mapping Gerät für das LoRaWAN TTN-Netzwerk auf Basis eines Arduino MKR WAN 1300

Bernd Laquai, 9. August 2018, V1

LoRaWAN und das „The Things Netzwerk“ (TTN)

Das LoRaWAN ist ähnlich wie das WLAN ein Funknetzwerk, mit dem ein Endgeräts Daten zu einem Gateway übertragen kann (vergleichbar zu einem WLAN-Router), der die Daten dann ins Internet weiterleitet. Allerdings lassen sich damit deutlich höhere Reichweiten im Kilometer-Bereich erreichen (Wide-Area-Network, WAN, im Gegensatz zum Wireless Local Area Network, WLAN). Daher muss nicht jeder Betreiber eines Endgeräts gleichzeitig auch ein Gateway aufstellen. Es reicht in den meisten Fällen, wenn sich ein Gateway, der von jemand anderem betrieben wird, in ein paar Kilometern Entfernung befindet.

Ein Ersatz für WLAN ist das LoRaWAN aber in den meisten Fällen nicht. Das LoRaWAN ist eine Low-Power Technologie für IoT (Internet-of-Things) -Anwendungen, wo typischerweise nur sehr geringe Datenmengen übertragen werden müssen. So ist beispielsweise die Überwachung des Wasserstands an einem Fluss eine typische IoT-Sensor-Anwendung für LoRaWAN. Dabei werden pro Stunde nur einige Bytes zum Gateway übertragen und das Sensor-Endgerät sollte einen möglichst geringen Leistungsbedarf haben, so dass beispielsweise auch ein Solarpanel die Stromversorgung übernehmen kann. Im Gegensatz dazu erfordert beispielsweise eine Übertragung der Daten einer HD-Webcam etliche Megabytes pro Bild. Dabei ist auch der Leistungsbedarf vergleichsweise hoch, so dass hierfür eher eine WLAN Funkverbindung sinnvoller ist. Das LoRaWAN kann dafür nicht benutzt werden.

Für die Weiterleitung der Daten in das Internet gibt es mehrere kommerzielle Anbieter aber auch freie Plattformen. Die in Europa bekannteste Plattform ist das „The Things Network“ (TTN), siehe auch https://www.thethingsnetwork.org/. Diese Plattform ist in den Niederlanden entstanden und breitet sich derzeit international aus. Sie wird hauptsächlich von Freiwilligen getragen, die Gateways aufstellen und diese über ein „Backend“ so vernetzen, dass die Integrität der Daten gewahrt bleibt und kein Durcheinander entsteht, wenn beispielsweise ein Datenpaket von mehreren Gateways empfangen wird. Dazu ist eine umfangreiche Server- und Software-Infrastruktur nötig, die vom TTN bereitgestellt wird. Anwender können sich auf der TTN Plattform registrieren, können Gateways und Endgeräte dort anmelden und sie in ihrem Anwenderbereich konfigurieren. Die Daten können dann entsprechend der Konfiguration über mehrere Wege ins Internet weitergeleitet werden, ein einfacher Weg ist z.B. der Versand mit dem HTTP-Internet-Protokoll.

Die LoRaWAN Funktechnologie nutzt das freie 868MHz Band zur Übertragung der Daten. Bei dieser Frequenz hat man noch eine gute Durchdringung von Mauerwerk und anderen Baustoffen. Dennoch merkt man im Gegensatz zu beispielsweise den Langwellen des Zeitzeichensenders DCF77 (77KHz) für Funkuhren, oder zu Kurzwellen, dass die Ausbreitung elektromagnetischer Wellen bei 868MHz auf Grund der hohen Frequenz bereits einen stark optischen Charakter hat. Wenn sich eine LoRaWan Antenne eines Endgeräts oder eines Gateways in einem Raum eines Hauses im Erdgeschoss befindet, bewegen sich die möglichen Reichweiten im Bereich von einigen 100m, stehen beide Antennen aber auf dem Dach und dazu noch auf hohen Gebäuden, nimmt die Reichweite massiv zu und kann bei freier Sicht mehrere 10km erreichen.

Konnektivität von LoRaWAN Endgeräten zu TTN-Gateways und die Mapper-Software

Wenn man sich zunächst drauf verlassen möchte, dass man für die eigene LoRaWan IoT-Anwendung einen fremden TTN Gateway nutzen kann, ist es nötig vorher die Konnektivität oder in anderen Worten die Empfangssituation zu testen. Einerseits braucht man dazu ein beispielhaftes Endgerät (manchmal auch „Knoten“ genannt) und muss dieses als registrierter Anwender auf der TTN Plattform anmelden. Zum anderen muss man die Empfangsmöglichkeiten für die von der TTN-Plattform empfangenen und per Internet zur Verfügung gestellten Daten in Abhängigkeit des Orts, an dem sich das Endgerät befindet, untersuchen, bevor man sich in Investitionen stürzt. Um diese Aufgabe zu vereinfachen, hat das TTN eine spezielle „Mapping-Software“ entwickelt, die als App auf einem Mobiltelefon läuft und über die üblichen App-Anbieter-Plattformen wie Playstore für Android heruntergeladen werden kann.

Diese Mapping-Software hat nun zwei Aufgaben, zum einen ermittelt sie über das GPS im Mobiltelefon den geografischen Aufenthaltsort des Telefons und geht dabei davon aus, dass sich auch das LoRaWAN Endgerät am gleichen Ort befindet. Die geographischen Daten werden von der Telefon-App an den TTN Server, der für das Mapping zuständig ist, gesandt. Zum anderen empfängt die Mapping-Software aber auch die von der Server-seitigen Software ermittelten Empfangsdaten zur Darstellung auf dem Mobiltelefon. So erhält der Anwender, der die App auf seinem Mobiltelefon am Laufen hat, während seiner Mapping Aktivität mit dem Endgerät eine live-Rückmeldung über die Empfangssituation an verschiedenen Orten oder auch entlang eines Wegs, den er zurücklegt. Die Daten, die von der Mapping-Software ermittelt werden, umfassen den Empfangszeitpunkt, die Anzahl der Gateways die ein Datenpaket des Endgeräts empfangen haben, sowie einen Indikator für die Empfangsfeldstärke (RSSI), das Signal-zu-Rausch Verhältnis am Gateway (SNR) und eine aus der Laufzeit abgeschätzte Entfernung des Gateways zum Ort des Endgeräts.

Abb. 1: LoRaWAN TTN-Mapper Gerät auf dem Fahrrad-Gepäckträger

Mit diesen Daten stellt die Mapping-App den geographischen Standort des Endgeräts für ein gesendetes Paket farblich codiert je nach der Empfangsstärke in einer Landkarte auf dem Mobiltelefon dar.

Genauso trägt der Mapping Server diese Daten in eine Datenbank ein und speichert sie. Aus den gespeicherten Datenbank-Einträgen können dann mittels einer Visualisierungs-Software auf dem TTN Server (https://ttnmapper.org/) die Ergebnisse vieler solcher Mapping Aktivitäten dargestellt werden, welche in der Summe einen Eindruck über die Netzabdeckung vermitteln.

Aufbau eines LoRaWAN Endgeräts für das Mapping

Ganz grundsätzlich kann jedes Endgerät (jeder LoRaWAN-Netzknoten) für ein Mapping benutzt werden. Die Mapping Software schaut bei einer Mapping Aktivität nicht auf den Inhalt der Daten, sie registriert nur den Empfang eines Pakets zusammen mit der Information über das Endgerät, welches das Paket versandt hat. Dennoch ist es ganz sinnvoll ein Endgerät so aufzubauen, dass es für eine Mapping Aktivität optimal geeignet ist. Dazu gehört, dass das Endgerät nur sehr wenige Daten sendet, diese aber beispielsweise alle 60 Minuten wiederholt. Damit kann sichergestellt werden, dass auch bei Bewegung entlang eines Weges eine gewisse Ortsauflösung erreicht wird, da dann auch genügend Pakete pro Wegabschnitt empfangen werden können.

Es ist zudem auch sinnvoll eine gute Antenne zu nutzen. Die Ende- und Empfangssituation eines LoRaWAN-Netzknotens hängt sehr stark von der verwendeten Antenne und der Antennen-Zuleitung ab. Gibt man sich hier nur wenig Mühe, dann ist die Konsequenz, dass das Mapping Ergebnis nicht die Netzabdeckung unter bestmöglichsten Bedingungen darstellt, sondern unter den Bedingungen einer schlechten Antenne (manchmal kann jedoch auch das gewünscht sein). Ein weiteres Augenmerk kann auf die Stromversorgung gelegt sein, da man unter Umständen eine Mapping-Aktivität über etliche Stunden hinweg durchführen möchte, so dass die Stromquelle eine ausreichende Kapazität aufweisen sollte. Soll eine Mapping Aktivität unter allen Wetterbedingungen möglich sein (Regen und Schneefall können Reichweiten erheblich reduzieren), muss auch darauf geachtet werden, dass das Gehäuse des Mapping Geräts ausreichend wasserdicht ist.

Das Arduino MKR WAN 1300 Board als Basis für ein Mapping Endgerät

Die LoRaWAN Technologie beruht auf einem von Semtech patentierten Modulationsverfahren (Chirp-Modulation). Daher gibt es zunächst die Semtech ICs, welche den LoRaWAN Standard erfüllen. In der Zwischenzeit hat Semtech dieses Patent aber auch an andere Halbleiter Hersteller lizenziert, z.B. an den chinesischen Hersteller HopeRF. Diese RF-ICs werden nun wiederum von etlichen Modul-Herstellern in sogenannte Funkmodule verbaut, darunter sind Hersteller wie Microchip, HopeRF, Murata, Embit, etc. Diese Funkmodule enthalten bereit oft schon dedizierte Mikrocontroller, welche nur das Funkprotokoll abdecken und nach außen eine einfache Datenkommunikation beispielsweise über einen SPI-Bus oder eine UART als elektrische Schnittstelle implementieren. Solche Module werden dann typischerweise von Herstellern auf Zusatzplatinen zu Mikrocontroller Plattformen oder aber direkt auf Mikrocontroller-Boards solcher Plattformen direkt verbaut.

Eine der bekanntesten großen Mikrocontroller Plattformen mit hoher Verbreitung ist die Arduino Plattform. Die italienische Firma Arduino, welche viele Mikrocontroller-Boards für unterschiedlichste Zwecke anbietet, hat auch eine ganze Familie an IoT-Boards entwickelt, die verschiedene Funkstandards mit on-Board Funkmodulen bedient. Sie wird MKR-Family genannt und basiert auf dem Atmel SAMD21 Mikrocontroller. Das Arduino Board aus dieser Familie, welches ein LoRaWAN Funkmodul enthält und damit auch eine Konnektivität zum TTN-Netzwerk ermöglicht, heißt MKR WAN 1300 und ist derzeit für rund 40 Euro zu haben. Dazu wird noch ein Antennenkabel und eine 868MHz Antenne benötigt, was zusammen bei etwa 10 Euro liegt.

Bei Antenne und Antennenkabel ist zu beachten, dass der MKR WAN 1300 eine Micro-U.FL Antennenbuchse auf dem Board hat. Nun gibt es zwei häufig verwendete 868MHz Antennentypen. Die

eine Sorte hat einen normalen männlichen SMA Stecker. Die andere Sorte hat einen „Reverse Polarity“ SMA-RP Stecker mit Außengewinde, der aber innen weiblich ist, d.h. das Antennenkabel muss dann als Innenleiter einen Stift haben. Daher ist es im Zweifel besser, die Antenne im Set mit Antennenkabel zu kaufen, damit die Kompatibilität von Anschlusskabel und Antenne sichergestellt ist. Grob kann man sagen, je größer der Durchmesser und je kleiner die Länge eines Antennenkabels ist, umso geringer ist auch seine zusätzliche Dämpfung. Allerdings setzt die Stecker-Größe schnell Grenzen, was den Durchmesser anbelangt. Ein Anschlusskabel mit geringer Dämpfung ist also vorzuziehen.

Eine Antenne mit einem Viertel der Wellenlänge als Länge ist von der Abstrahlungseigenschaft und der Baugröße her ein guter Kompromiss, d.h. die Antenne sollte nicht kürzer als 7 cm sein sonst ist sie intern „elektrisch verlängert“, was nicht so günstig ist.

Wichtig ist auch noch, dass beim Betrieb des Boards immer eine Antenne als „Hochfrequenzlast“ angeschlossen sein sollte um eine Beschädigung des Funkmoduls auszuschließen.

Abb. 2: MKR WAN 1300 mit Antenne

Programmierung des MKR WAN 1300

Der MKR WAN 1300 wird unter der Arduino Entwicklungsumgebung programmiert. Dazu sollte die neueste Version benutzt werden, da das Board relativ neu ist. Diese Software kann auf der Webseite https://www.arduino.cc/ unter →Software→Downloads heruntergeladen werden. Eine detaillierte Anleitung zur Programmierung des MKR WAN 1300 ist auf https://www.arduino.cc/ unter →Products→Arduino→MKR WAN 1300 (Internet-of-Things) →Getting Started zu finden. Auf dieser Seite findet man unter den Tutorials eine Anleitung „How to connect MKRWAN 1300 to The Things Network (TTN). Dort sind auch die wichtigsten Schritte für die Anmeldung des MKR WAN 1300 „Device“ (Endgerät) auf dem Portal des TTN beschrieben.

Dieser Anleitung folgend muss also zunächst das „First Configuration Sketch“ Beispiel aus der MKRWAN library ausgeführt werden, damit der Funkbaustein seine intern gespeicherten Daten preisgibt, mit denen das „Device“ bei TTN angemeldet werden kann.

Wenn diese Anmeldung vollständig ist, kann der MKR WAN bereits als Endgerät betrieben werden. Um den MKR WAN als Mapper einzusetzen, empfiehlt es sich das LoraSendandReceive Beispiel aus der MKRWAN Bibliothek ein wenig abzuändern, so dass währen des Betriebs keine Eingaben gemacht werden müssen und beispielsweise nur der String „ABC“ als Datenmessage verschickt wird (siehe das Listing im Anhang zu diesem Dokument).

Beim Hochladen des kompilierten Sketches könnte im Message Fenster der aktuellen Software Version noch die Fehlermeldung „SAM-BA operation failed“ erscheinen. Diese Meldung kann offensichtlich ignoriert werden, denn der Code wird dennoch hochgeladen und richtig ausgeführt (Arduino Version 1.8.5). Es kann auch nötig sein, dass zweimal kurz auf den Reset-Knopf gedrückt werden muss, damit die Entwicklungsumgebung nach dem Hochladen wieder auf den richtigen Port zurückspringt. Dies scheint auch noch ein kleines Problem bei der neuen Software zu sein.

Aufbau des Endgeräts mit dem MKR WAN 1300 als Mapper

Die oben beschriebene Programmierung und der erste Betrieb als „Device“ (Endgerät) im TTN Portal kann noch gut mit einem Micro USB Kabel als Verbindung zwischen Laptop und dem MKR WAN 1300 Board erfolgen. Danach aber ist es zweckmäßig das Mikrocontroller Board in ein einigermaßen wasserdichtes Gehäuse fest zu montieren und mit einem eigenen Akku für den autarken Betrieb zu versehen.

Als Stromversorgung des MKR WAN 1300 wird am günstigsten ein 7.4V LiPo Akku aus dem Modellbau verwendet, dessen Ausgangsspannung über einen DC-DC Konverter-Baustein stromsparend auf 5V heruntergeregelt wird. Der Vorteil ist dann, dass sich die Akkuspannung dann über einen weiten Bereich ändern kann, die 5V aber stabil bleiben. Ein LiPo-Akku kann über günstige Ladegeräte aus dem Modellbau einfach wieder aufgeladen werden. Als DC-DC Baustein eignet sich z.B. der Typ TSR 1-2450 von Traco Power, der von den meisten Elektronik-Bauteile-Distributoren bezogen werden kann. Der Pluspol des Akkus wird über einen Ein/Aus-Schalter am Uein-Anschluss des dreipoligen Gehäuse des DC-DC Konverters am Pin angeschlossen. Die auf 5V geregelte Ausgangs-Spannung Uaus muss dann am MKR WAN über den Pin Vin eingespeist werden, da sie auf dem Board noch einmal auf 3.3V herunter geregelt wird. Der Minuspol des Akkus und der Ground Anschluss GND am MK WAN Board muss mit dem Ground Anschluss am DC-DC Konverter verbunden werden. Mehr an Verdrahtung ist nicht nötig. Lediglich die Antenne muss noch angeschlossen werden. Abb. 3 zeigt das Anschluss-Schema.

Als Montageplatte für die Montage des MKR WAN Boards in einem Gehäuse kann eine Lochrasterplatine verwendet werden. Der MKR WAN 1300 wird bereits mit eingelöteten „stackable headers“ ausgeliefert. Das sind Buchsenleisten, die nach unten gleichzeitig Stiftleisten sind. Diese können wiederum in Buchsenleisten gesteckt werden, die lediglich zur Fixierung auf die Lochrasterplatine montiert werden. Für eine elektrische Kontaktierung werden sie nicht benötigt, da die Stromversorgung auch von oben in die Buchsenleiste des Arduino-Boards eingesteckt werden kann. Zudem kann das DC-DC Modul einfach auf die Platine gelötet und Anschlusskabel angelötet werden, das dann wiederum über den Schalter mit dem Akku bzw. mit dem MKR WAN Board verbunden wird.

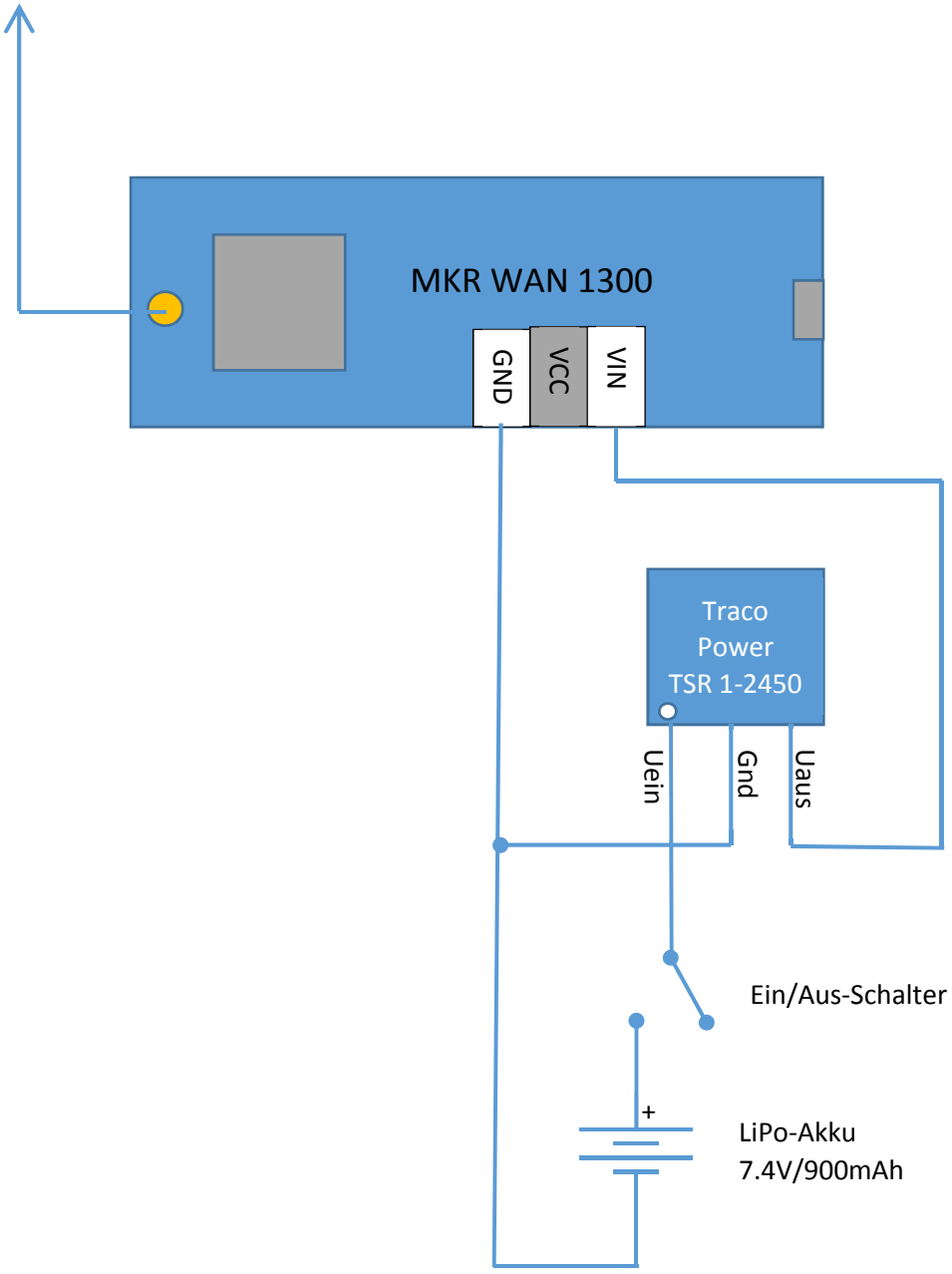

Abb. 3: Schaltplan für den TTN-Mapper

Auf der Platine findet dann auch der Akku Platz (fixiert mit einem Kabelbinder), der sicherheitshalber über den Balancer-Ladeanschluss angeschlossen wird. Ein 7.4V hat 2 LiPo Zellen (2S) und daher 3 Balancer Anschlüsse. Der mittlere Anschluss führt zwischen die zwei Zellen und erlaubt es dem Ladegerät, dass die Zellen unabhängig voneinander zu laden. Das Laden über ein solches Balancer-Ladegerät erhöht die Lebensdauer. Die beiden äußeren Anschlüsse am Balancer-Stecker sind +7.4V und Gnd. Nur diese werden für den Mapper benutzt, der mittlere Anschluss bleibt frei. Die Hochstromanschlussbuchse verklebt man am besten mit einem Isolierband kurzschlusssicher. Jede andere Batterie bzw. Akku mit einer Spannung zwischen 6.5V und 24V ist natürlich genauso als

Stromversorgung denkbar, der DC-DC regelt in diesem Bereich immer auf 5V am Ausgang und kann 1A Strom liefern. Soviel wird natürlich bei weitem nicht benötigt. Ganz im Gegenteil, der MKR WAN 1300 ist extrem stromsparend und läuft etliche Stunden mit einem kleinen LiPo Akku. Eine 3V Versorgung mit AA Batterien über den grünen Stecker, wie von Arduino angegeben, ist dagegen etwas knapp, da die ICs eigentlich für 3.3V Soll-Spannung spezifiziert sind.

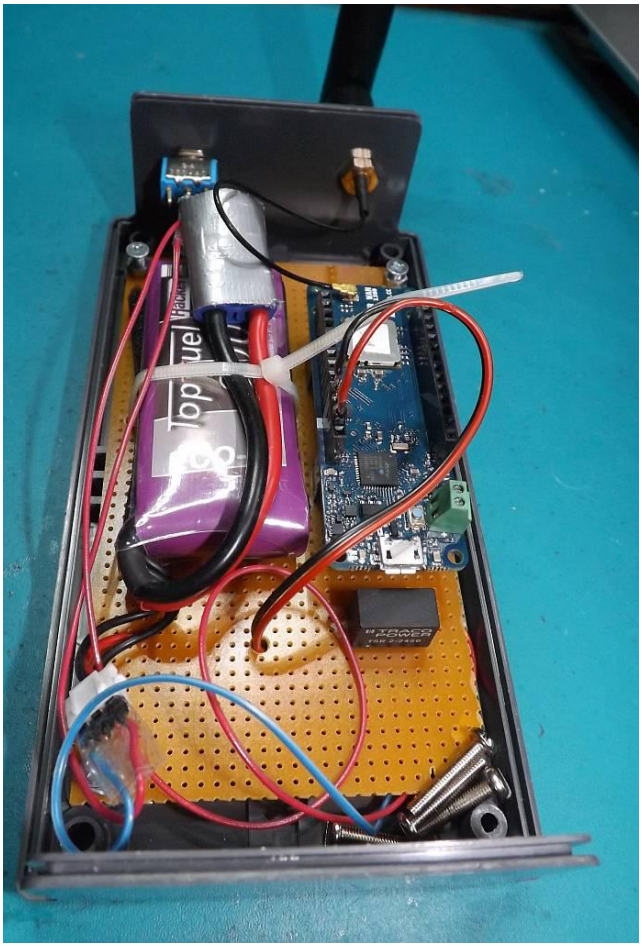

Abb. 3: Innenaufbau des TTN-Mappers

Inbetriebnahme des TTN-Mappers

In dem Augenblick, in dem das Gerät eingeschaltet wird, beginnt dann der programmierte MKR WAN 1300 Datenpakete mit der Message „ABC“ im Minutenabstand zu versenden. Dies kann über die TTN-Konsole (der TTN-Anwenderbereich) kontrolliert werden. Wenn man unter Applications→<Anwendungsname>→Devices→<Gerätename>→Data auf die Webseite schaut, werden dort unter Payload die Daten 41 42 43 angezeigt, da sind die ASCII-Codes für A B C dargestellt in hexadezimaler Form.

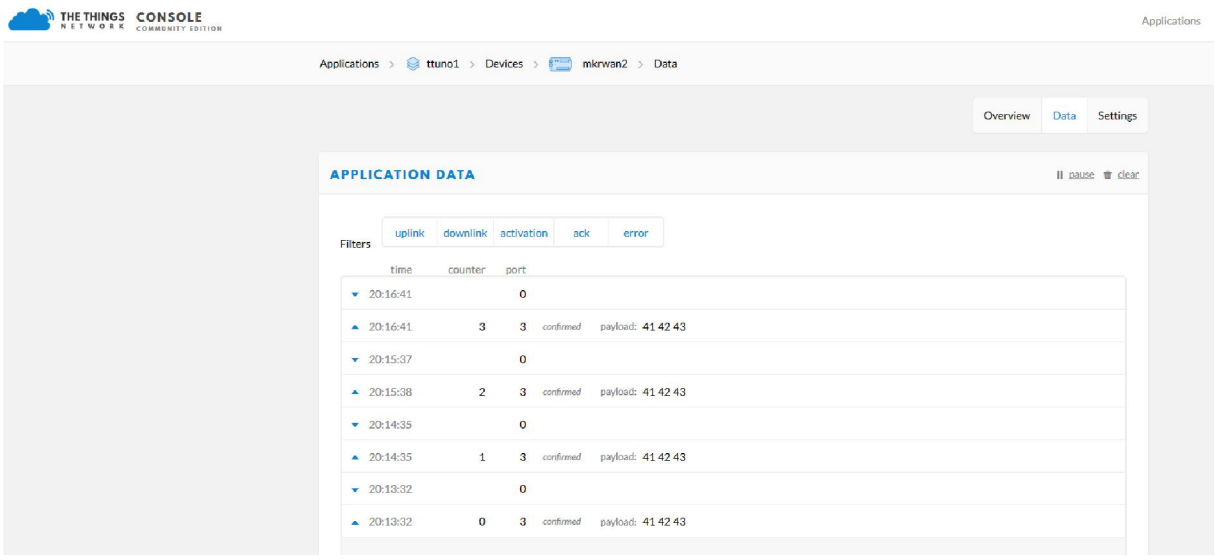

Abb. 4: Der TTN-Konsolenbereich für die Anzeige der Daten des Mapper-Geräts

Wenn dieser Test erfolgreich war, dann kann die kostenlose TTN Mapper App heruntergeladen und installiert werden.

Ruft man die App auf, befindet sich die Software zunächst in einem Standby Modus, was man an der Zeile „Ready to start mapping.“ links oben erkennen kann. Gleichzeitig wird eine Weltkarte angezeigt und das GPS im Telefon aktiviert. Drückt man auf den blauen Softbutton mit den drei vertikalen weißen Punkten links unten, kommt man ins Konfigurationsmenu. Dort gibt es einen blauen Button „Settings“ welcher auf eine Seite führt die eine Auswahl des Geräts ermöglicht, das zum Mapping verwendet werden soll. Das hier ausgewählte Gerät kann jederzeit geändert werden.

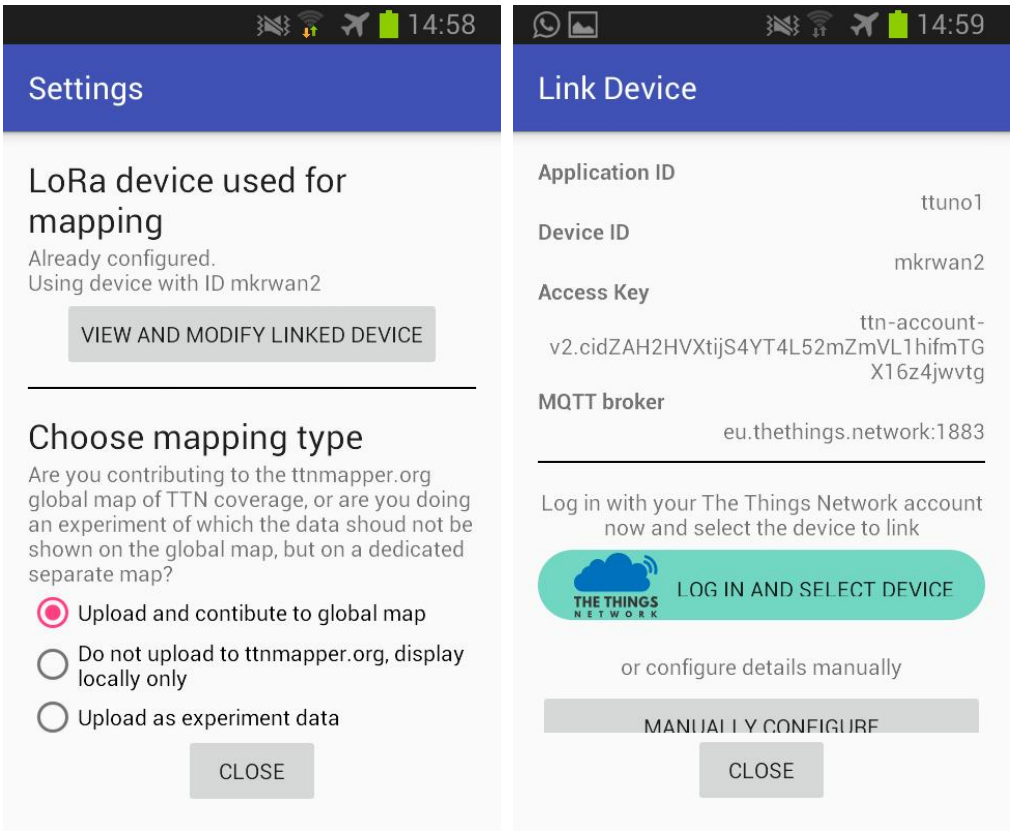

Abb. 5: Konfiguration der TTN Mapper App

Es ist zweckmäßig, sich über die App in dem eigenen Konsolenbereich einzuloggen und so das gewünschte Gerät (Device) mit seinem Anmeldename auszuwählen. Dieses wird dann dem Mobiltelefon mit der Mapping-App zugeordnet.

Sobald da GPS den Standort ermittelt hat, kann Der Mapping Vorgang mit dem Button „Mapping“ rechts unten auf der Landkarte gestartet werden. Hat man in der Konfiguration „Auto Center“ und „Auto-Zoom“ aktiviert, dann fokussiert die Software die Karte nun auf den Standort und meldet links oben „Mapping started.“. Hat man eine Prepaid SIM-Karte im Telefon schaltet man danach am besten diese beiden Optionen wieder aus, um das beliebige Nachladen von Kartensegmenten zu unterbinden, da dies höhere Telefonkosten erzeugt. Dann muss man aber selbst dem Standort durch Schieben und Zoomen der Karte folgen.

Sobald die Mapper App den Empfang von Paketen durch Gateways in der Nähe gemeldet bekommt, wird dies auf der Karte durch farblich codierte Punkte am jeweiligen Standort angezeigt.

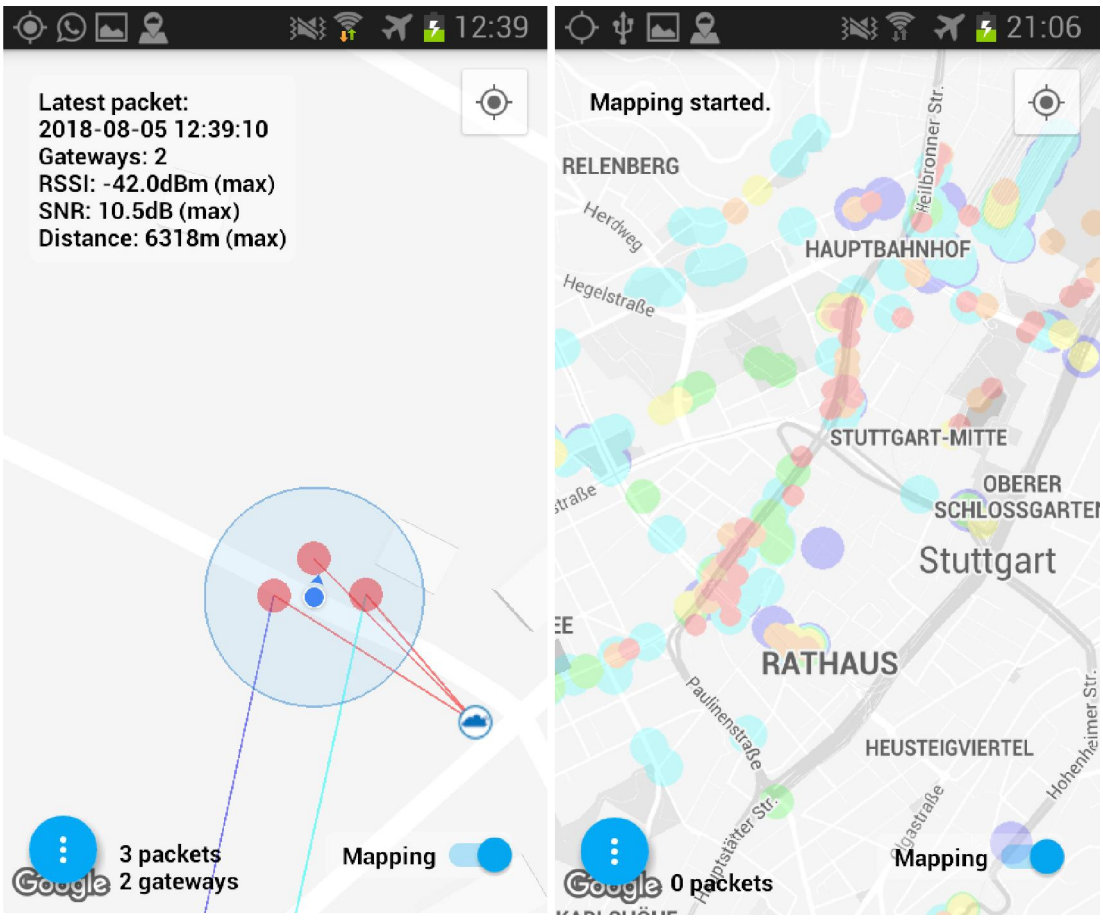

Abb. 6: Darstellung der Mapping Ergebnisse auf dem Mobiltelefon

Rote Punkte bedeuten sehr guten Empfang, typischerweise in der direkten Umgebung eines Gateways. Von den Punkten werden auch Linien angezeigt zu den Gateways, die ein Paket empfangen haben. Das können auch mehrere Gateways gleichzeitig sein. Um den empfangenden Gateway anzuzeigen, werden farblich nach der Empfangsstärke entsprechend codierte Linien vom jeweiligen Empfangsstandort zum Gateway hin in die Karte eingezeichnet. Die Daten des am weitesten entfernten Gateways werden zusätzlich oben am Bildschirm als Text eingeblendet.

Wenn man in der Konfiguration den Button „Coverage“ angewählt hat, dann werden auf dem Server bereits gespeicherte Mapping Ergebnisse halb-transparent dargestellt. Damit bekommt man einen

Überblick, welche Gebiete noch nicht gemappt wurden bzw. wo welche Empfangsstärke zu erwarten ist.

Hat man in der Konfiguration als Mapping-Type „Upload and contribute to global map" angewählt, dann werden die selbst erzeugten Mapping Daten auf dem TTN Server gespeichert und ebenfalls zur Kartenerstellung für die Gemeinschaft genutzt. Die Visualisierung dieser Daten kann man sich unter https://ttnmapper.org/ auf dem Mapper-Portal im Internet nach einer gewissen Zeit in verschiedenen Variationen anschauen.

Die eigenen Mapping-Aktivitäten tragen also mit dazu bei, dass die derzeitige Netzabdeckung ermittelt werden kann und helfen verschiedenen freiwilligen Organisationen und Hobbyisten bei der Planung der Aufstellungsorte für weitere Gateways. So gesehen ist das Mapping ein äußert hilfreicher Beitrag der Anwender für die gesamte TTN-Community. Ganz nebenbei macht das Mapping aber auch einfach Spaß und gibt einem einen gewissen Eindruck der maximal möglichen Reichweiten und des Ausbreitungsverhaltens der Funkwellen im freien 868MHz Band.

Happy Mapping.

Anhang

Sketch Listing für den MKR WAN als TTN Mapper Device

```
/*
  Modified Lora Send And Receive Example for use as mapper device
*/

#include <MKRWAN.h>

LoRaModem modem;

// Uncomment if using the Murata chip as a module
// LoRaModem modem(Serial1);

//#include "arduino_secrets.h"
// Please enter your sensitive data in the Secret tab or arduino_secrets.h
String appEui = "<myappEUI>";
String appKey = "<myAppKey>";


void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  //while (!Serial); // mit diesem Statement startet das Board nicht ohne Serieal Monitor!!!

  pinMode(LED_BUILTIN, OUTPUT);        //sign of live
  for (int i=1; i<=10; i++) {
    digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(1000);                       // wait for a second
    digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
    delay(1000);
    // wait for a second
    Serial.println(10-i);
  }

  // change this to your regional band (eg. US915, AS923, ...)
  if (!modem.begin(EU868)) {
    Serial.println("Failed to start module");
    while (1) {}
  };
  Serial.print("Your module version is: ");
  Serial.println(modem.version());
  Serial.print("Your device EUI is: ");
  Serial.println(modem.deviceEUI());
```

```
  int connected = modem.joinOTAA(appEui, appKey);
  if (!connected) {
    Serial.println("Something went wrong; are you indoor? Move near a window and retry");
    while (1) {}
  }

  // Set poll interval to 60 secs.
  modem.minPollInterval(60);
  // NOTE: independently by this setting the modem will
  // not allow to send more than one message every 2 minutes,
  // this is enforced by firmware and can not be changed.
}

void loop() {
  //Serial.println();
  //Serial.println("Enter a message to send to network");
  //Serial.println("(make sure that end-of-line 'NL' is enabled)");

  //while (!Serial.available());
  //String msg = Serial.readStringUntil('\n');
  String msg = "ABC";

  Serial.println();
  Serial.print("Sending: " + msg + " - ");
  for (unsigned int i = 0; i < msg.length(); i++) {
    Serial.print(msg[i] >> 4, HEX);
    Serial.print(msg[i] & 0xF, HEX);
    Serial.print(" ");
  }
  Serial.println();

  int err;
  modem.beginPacket();
  modem.print(msg);
  err = modem.endPacket(true);
  if (err > 0) {
    Serial.println("Message sent correctly!");
  } else {
    Serial.println("Error sending message :(");
    Serial.println("(you may send a limited amount of messages per minute, depending on the
signal strength");
    Serial.println("it may vary from 1 message every couple of seconds to 1 message every
minute)");
  }
  delay(1000);
  if (!modem.available()) {
    Serial.println("No downlink message received at this time.");
  }
  else {
    String rcv;
    rcv.reserve(64);
    while (modem.available()) {
      rcv += (char)modem.read();
    }
    Serial.print("Received: " + rcv + " - ");
    for (unsigned int i = 0; i < rcv.length(); i++) {
      Serial.print(rcv[i] >> 4, HEX);
      Serial.print(rcv[i] & 0xF, HEX);
      Serial.print(" ");
    }
    Serial.println();
  }
  Serial.println("waiting 60 seconds");
  for (int i=1; i<=60; i++) { //wait a minute
    Serial.println(i);
    delay(1000);
  }
}
```