



Willkommen

zum

LoRaWAN – Basteln mit dem Arduino

Technologiestiftung Berlin, 17.01.2019

Dr. Christian Hammel

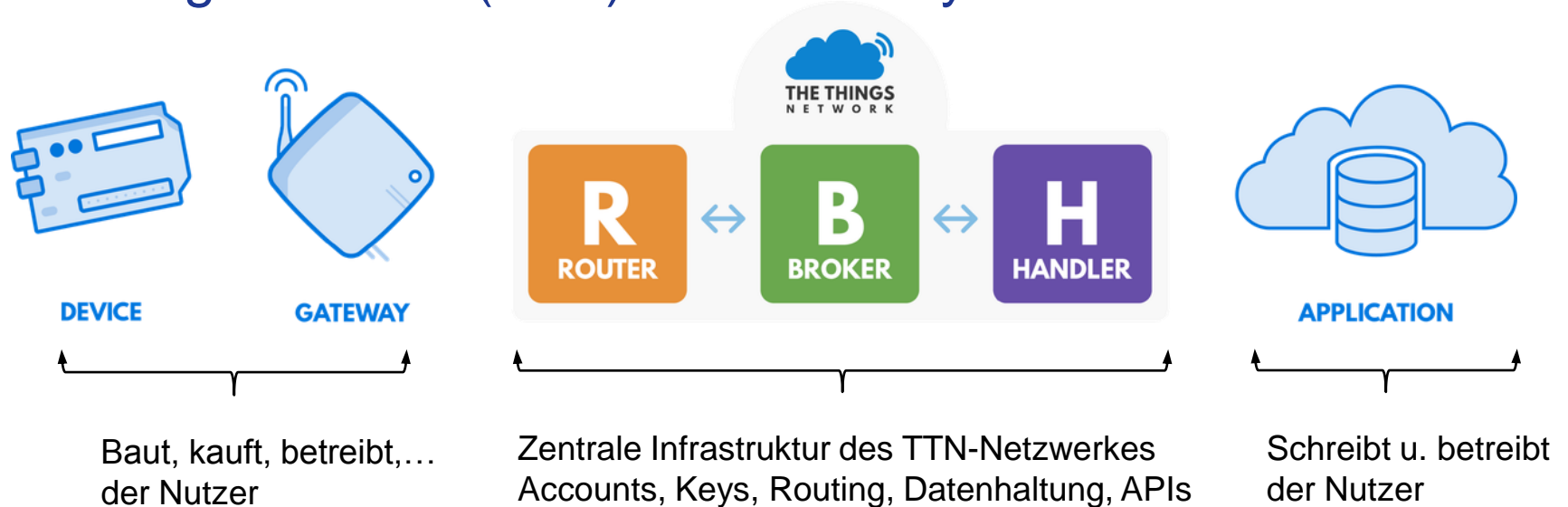
Carolin Clausnitzer

Dr. Benjamin Seibel

Lemuel Lakritz



TheThingsNetwork (TTN) - Community-Netz



Die zentrale Struktur kann jeder kostenlos nutzen. Sie bietet Geräte-Keys, Nutzerverwaltung, Paketrouting, End-zu-End-Verschlüsselung (AES-128) und API (HTTP, MQTT).

Kostenpflichtig: Leistungen der TTN Industries wie private Server, Integration in Kundensysteme,...

LoRaWAN-Sprech: Device = Mote = Node = Endgerät; Gateway = Basisstation (= Router)

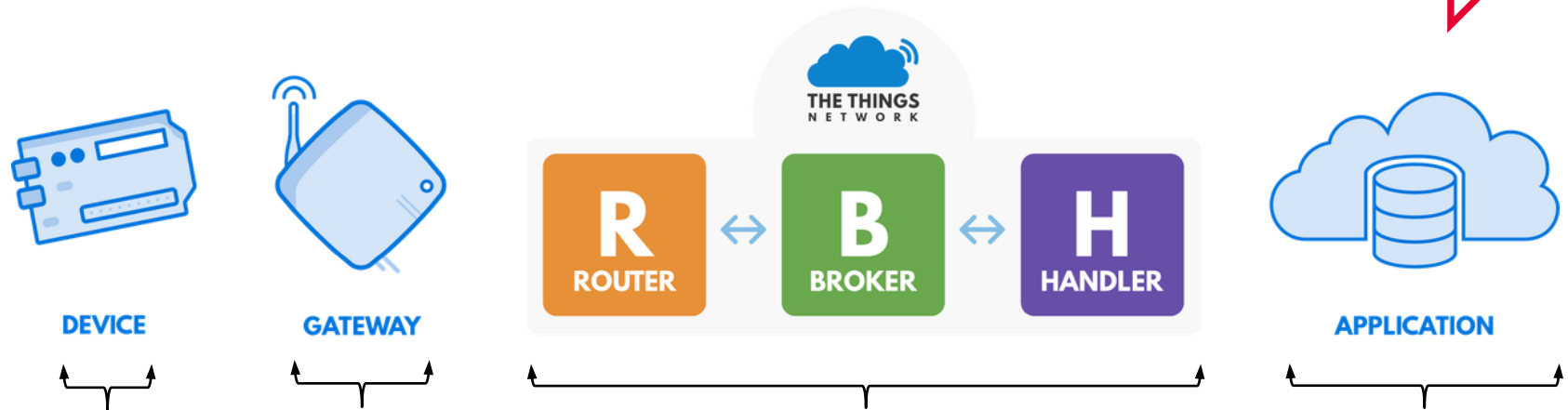
Funkregulierung: 1% Airtime, also Sendeintervalle nicht beliebig verkürzen!

Stand:

- 500 Communities weltweit, 42.000 User, 4.000 Gateways, 20.000 Applications, Schwerpunkte NL, BE, CH, UK
- TTN-Berlin: 60 Contributors, 45 Gateways

Worum geht es heute? Was machen wir?

Wir übertragen selbst gemessene Daten vom Arduino ins Netz (uplink)



Unser
Arduino!

Vorhanden

Das benutzen wir einfach.

Hier spielt die Musik,
aber heute nicht.

Wir schließen einen Sensor an.
Wir messen eine Temperatur.
Wir melden das Gerät bei TTN an.
Wir senden die Temperatur ins TTN.
Wir nutzen Beispielcode.

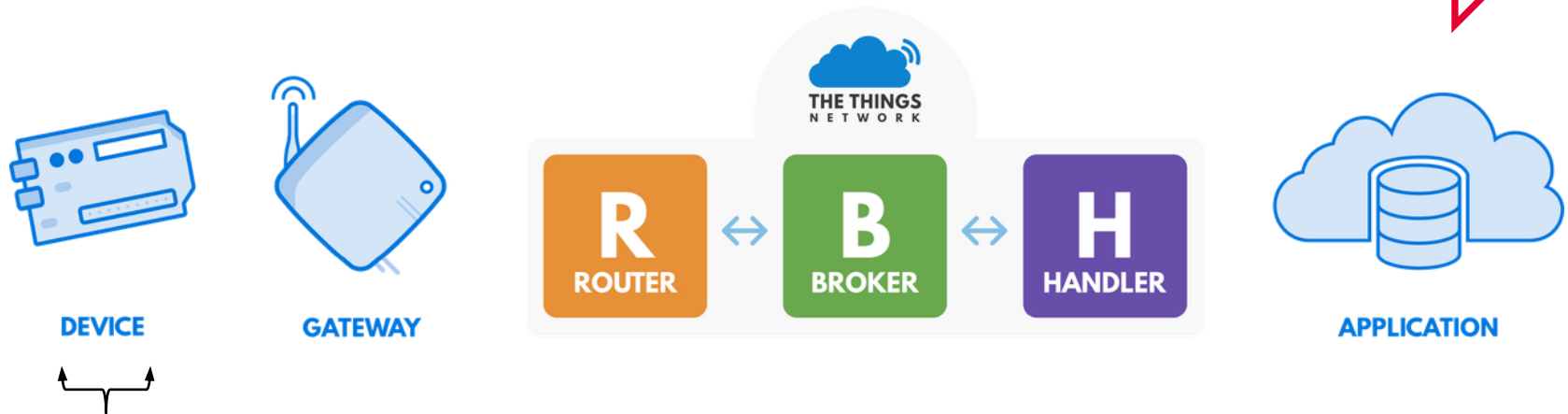
Teilnehmer sollten einen
Account haben, damit sie ihr
node anmelden und auf ihre
Daten zugreifen können.

Wie melde ich den Arduino an?
Wo sehe ich meine Daten?

Wenn wir Zeit haben,
lesen wir unsere
Daten mit einem
MQTT-Viewer.

Was genau soll unser Device machen?

Wir übertragen selbst gemessene Daten vom Arduino ins Netz (uplink)



Unser Arduino

- Misst die aktuelle Temperatur mit einem Sensor
- Überträgt die Temperatur (genauer: Bytes) ins TTN (genauer: zum TTN-Broker)
- Optional: Blinkt mit einer Kontroll-LED, wenn die Temperatur über 30° ist, damit wir direkt sehen können, ob unsere Schaltung und unser Code funktionieren.

TTN

- Zeigt uns in der Ansicht „Console“ unsere Temperatur an (wenn wir die Bytes in brauchbare Daten zurückverwandelt (decodiert) haben)

Application MQTT-Viewer

- Zeigt die Daten im Browser oder auf dem Smartphone an, wenn noch Zeit ist

Board zusammenbauen

1. **LoraRaWAN-Shield auf den Arduino montieren**

(so aufstecken, dass er gewaltfrei passt und dass die Anschlüsse auf Arduino und Shield in Deckung sind)

2. **ANTENNE ANSCHRAUBEN !!!!!**

(NIE ohne Antenne betreiben, das kann das Board zerstören)

Arduino-Code

1. **Arduino-Code:** Die IDE kompiliert den c-ähnlichen Code in Maschinencode, der dann auf den Arduino geladen wird. Deshalb kann man Code auch nicht vom Arduino herunterkopieren.
2. **Programmaufbau:** Es gibt immer mindestens drei Teile,
 - einen Header, in dem includes, Variablen u.ä. definiert werden,
 - einen Teil void setup (){}, der genau einmal abgearbeitet wird und
 - einen Teil void loop (){}, der ständig wiederholt wird.Man kann an beliebiger Stelle eigene Funktionen ergänzen, die sich aus anderen Programmteilen heraus aufrufen lassen.
3. **Syntax**
Funktionen: void FUNKTIONSNAME (Rückgaben) {Code der Funktion}
Befehlszeilen: enden immer mit ;
Kommentar: wird mit // eingeleitet und nicht kompiliert
Konstanten und Variablen: Sind immer 2 Byte (16 Bit), wenn man nichts anders definiert.
4. **Progrämmchen (Sketches)**
Konvention: enden auf .ino
stehen in einem Ordner namens Arduino-Sketchbook in eurem Userverzeichnis.
5. **Beispiele für heute:** <https://github.com/technologiestiftung/workshops>

Kommunikation Lappi / Arduino testen

1. Arduino-IDE starten

2. Arduino über USB-Kabel anschließen

(NIE ohne Antenne!)

3. Kommunikationstest

„Werkzeuge | Board | Mega“, „Werkzeuge | Prozessor | Mega2560“; „Werkz. | Port | \$richtigerPort“;

„Werkzeuge | Boardinformationen holen“

Wenn etwas kommt: alles richtig angeschlossen,

Wenn nicht oder Fehler: richtiger Port? Strom am Arduino? USB-Anschluss freigegeben (Linux)?

4. LMIC (LoRaWAN-Bibliothek einbinden)

„Sketch | Bibliothek einbinden | IBM LMIC framework“

5. Testcode hochladen

Testcode der TSB (kommunikationstest_lappi_arduino.ino) in die IDE laden; „Sketch | hochladen“

Keine Fehlermeldungen? Alles prima!

Dragino mit GPS und Fehlermeldung in diesem Stil:

```
avrduede: stk500v2_ReceiveMessage(): timeout
```

```
avrduede: stk500v2_getsync(): timeout communicating with programmer
```

GPS-Reset-Knopfchen während Hochladen gedrückt halten

6. Test

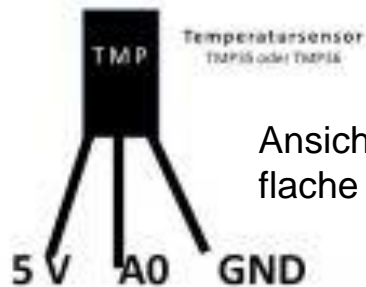
„Werkzeuge | serieller Monitor“:

Wenn bei 115200 baud sinnvoller Output kommt, ist alles prima.

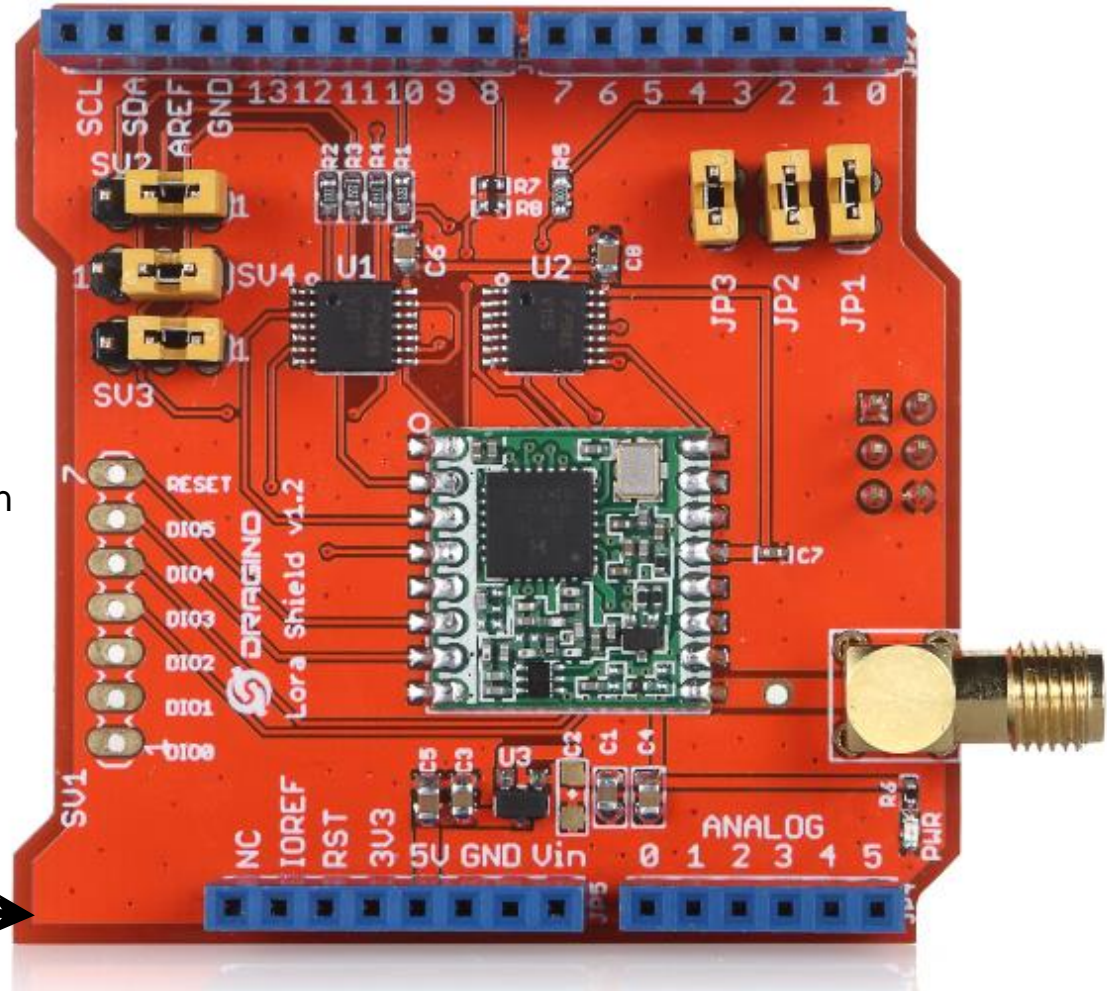
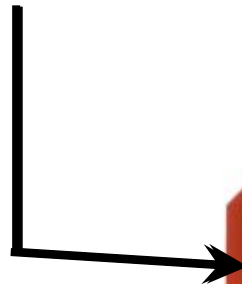
Schaltungen aufbauen

Sensor am Arduino anschließen:

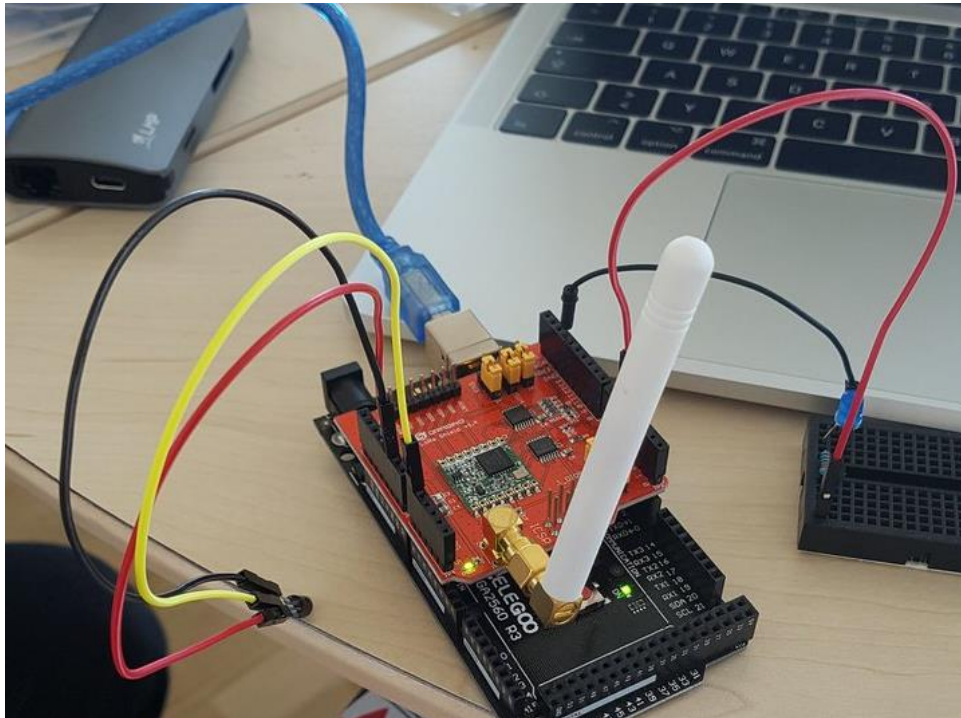
- GND an GND (Analogseite)
- 5V an 5V (Analogseite)
- Messausgang auf Analogeingang 0



Ansicht:
flache Seite oben

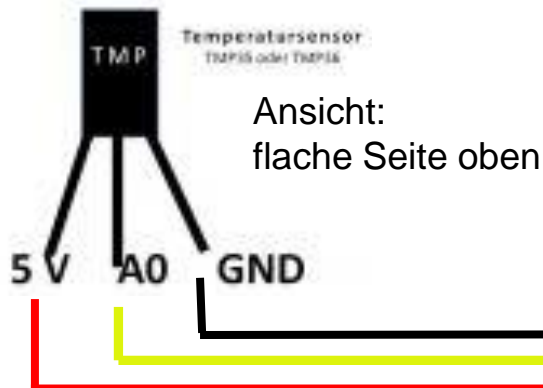
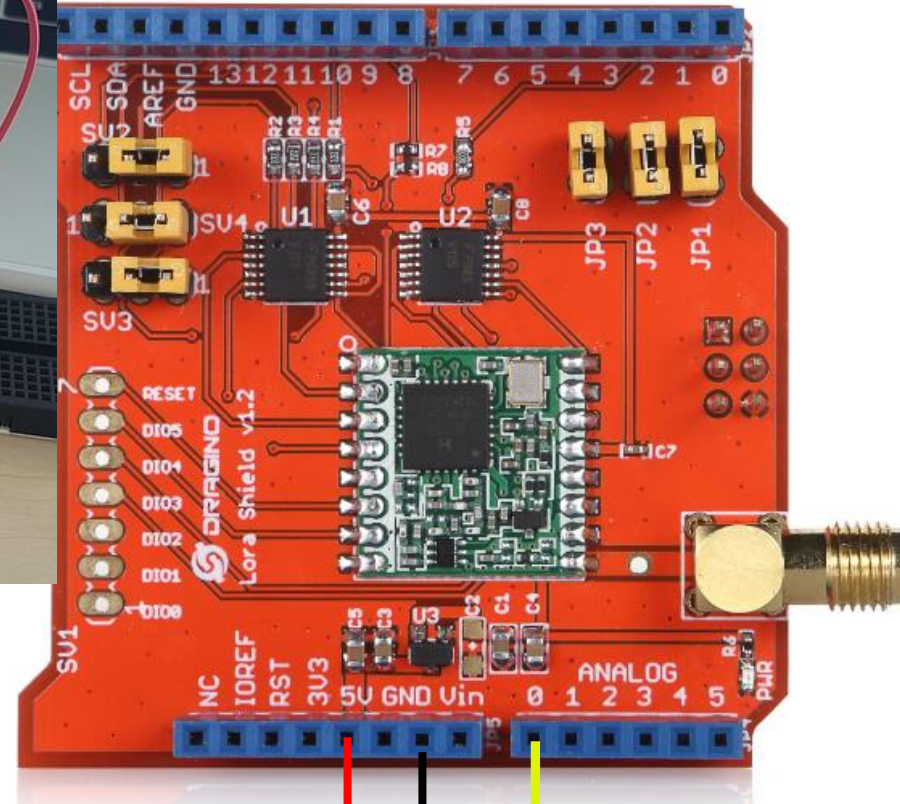


Schaltungen aufbauen: Sensor



Sensor am Arduino anschließen:

- GND an GND (Analogseite)
- 5V an 5V (Analogseite)
- Messausgang auf Analogeingang 0



Arduino bei TTN anmelden, Daten senden und decodieren

1. Bei TTN einloggen (wer keinen Account hat, muss sich einen anlegen)

TTN-Konsole:

Anwendung anlegen,

device anlegen,

Settings, ABP, generiert Schlüssel und Device-ID

2. Arduino IDE starten

TSB-Mustercode (temperatur_basteltreff.ino) in die IDE **aber noch nicht auf den Arduino** laden (die Schlüssel im Beispiel sind fake und produzieren Fehler).

Die Schlüssel Network Session Key, Appskey und Device ID (\neq EUI !) in den Mustercode einpflegen (als hex, wird mit <-> erzeugt und mit dem „Auge“ kopiert).

Eine beliebige interne ID könnt ihr einbauen, wenn ihr mehrere Nodes auseinanderhalten wollt.

3. Code hochladen

Wenn eure Schlüssel richtig eingetragen sind, dann seht ihr im seriellen Monitor Daten, die ihr erwartet und in eurer Konsole Daten, die noch nicht sinnvoll sind.

Wenn die optionale Kontroll-LED angeschlossen ist und der Sensor $>30^\circ$ warm ist, geht die LED an.

4. Payload-Decoder einrichten

Damit die an TTN gesendeten Daten wieder in eine sinnvolle Form kommen, legt ihr über die TTN-Konsole einen payload-decoder an. Code dafür (javascript) steht als Kommentar im Arduino-Testcode.

5. Code verstehen

versuchen wir direkt im Code, dafür sind Kommentare im Code.

Daten ansehen und prüfen

1. Direkt bei TTN

TTN-Konsole: Bei eurer Application / Eurem Device könnt ihr die Daten live sehen und wenn der Decoder funktioniert, werden sie auch entschlüsselt.

2. Von TTN auslesen

über **http**: geht, sprengt hier aber schnell den Rahmen

über **MQTT**: für diesen Standard gibt es einige fertige Viewer.

3. MQTT

liefert Datenfelder, die ein Viewer „abonniert“ und anzeigt, nutzt üblicherweise tcp auf Port 1883.

Ein Datenfeld hat Name und Wert und heißt in MQTT-Sprech „Topic“.

Da wir eine End-zu-End-Verschlüsselung haben und auf das Application-Ende nicht jeder Zugriff haben soll, braucht der Viewer einen Schlüssel/password zum Zugriff auf die Daten.

Unsere Topics sehen so aus: **AppID/devices/deviceID/up (/EUERDATENFELD)**

Wenn ihr viele devices habt, sollte das so aussehen: AppID/devices/+/up (/EUERDATENFELD)

Zugang: Host: eu.thethings.network ; Port: 1183 ; Username; App-ID ; Password: App-Access Key

4. MQTT mit Android-Apps: MQTT Dash, Linear MQTT Dashboard, MyMQTT

(Tipp: App-Access-Key abtippen geht immer schief!

Lösung: Browser oder die App TTN Mobile können copy und paste.)

5. MQTT mit Chrome: Plugin MQTT-Lens

Daten nutzen: Live-Graph mit LinearMQTT (Android)

1. Sicherheit und Privacy

Ihr installiert eine App und gewährt ihr Rechte. Der Hersteller „ravendmaster“ ist lt. Impressum in Russland .

Wem das zu dubios ist, der sollte es nicht nutzen.

2. MQTT-Verbindung herstellen

„Menu“-Symbol oben rechts; App-Settings

Server: tcp://eu.thethings.network ; Port: 1883 ; Username: AppID ; User password: App-Access Key (den zeigt die TTN-Konsole an); Client ID: generiert die App selbst

(Tipp: App-Access-Key abtippen geht immer schief! Browser oder TTN Mobile können copy und paste)

3. Graph-Widget basteln

„Menu“-Symbol oben rechts; Tabs ; „+“ ; Tabnamen vergeben; zum neuen Tab wechseln; „+“-Symbol legt ein Widget an; Widget-Type: „Graph“; „Live“; „Name“: beliebig, wird später so angezeigt; Sub.Topic: : AppID/devices/deviceID/up/EUERDATENFELD ; SAVE

EUERDATENFELD heißt Temperatur, wenn ihr unser Beispiel nicht geändert habt

Dieses Widget zeigt Eure Temperatur Live als Graph an.

Daten nutzen: blinkendes Widget mit MQTTDash (Android)

1. Sicherheit und Privacy

Ihr installiert eine App und gewährt ihr Rechte. Der Hersteller „routix.net“ hat kein Impressum, auch keinen adminC im whois. Die Domain ist in Russland registriert, dort steht auch der Server (tracert).

Wem das zu dubios ist, der sollte es nicht nutzen.

Entzug aller Berechtigungen, die man mit Android6 entziehen kann: funktioniert trotzdem.

2. MQTT-Abonnement anlegen

„+“-Symbol oben rechts; „Name“: beliebig, wird später so angezeigt; Address: eu.thethings.network ; Port: 1883 ; Username: AppID ; User password: App-Access Key (den zeigt die TTN-Konsole an); Client ID: generiert die App selbst

(Tipp: App-Access-Key abtippen geht immer schief! Browser oder TTN Mobile können copy und paste)

3. Widget basteln

Name: beliebig (so heißt euer Widget dann), Topic: AppID/devices/deviceID/up/EUERDATENFELD
EUERDATENFELD heißt Temperatur, wenn ihr unser Beispiel nicht geändert habt.

Folgefelder: Enable, Update, Large, QoS(0),(Blink: val > 29)

Daten ansehen mit MQTT-Lens (Chrome-Plugin)

1. Sicherheit und Privacy

Ihr installiert eine Chrome-Erweiterung App und gewährt ihr Rechte! Die privacy von Chrome ist umstritten. **Wem das zu dubios ist, der sollte es nicht nutzen.**

2. **Leistung:** MQTT-Lens zeigt eure Datenfelder vollständig an (nicht sehr schön aber lesbar), schicke Visualisierungen kann es nicht, außerdem startet es auf älterer Hardware sehr langsam und stürzt gerne mal ab. Dafür kann man damit über MQTT auch Messages zum device senden.
Achtung: geht nur, wenn der Port 1883 nicht gesperrt ist!

3. MQTT-Verbindung anlegen

„**Connections** | „+“-Symbol; „Name“: beliebig, wird später so angezeigt; Hostname: tcp, eu.thethings.network ; Port: 1883 ; Client ID: generiert die App selbst; Username: AppID ; User password: App-Access Key (den zeigt die TTN-Konsole an);
(Tipp: App-Access-Key abtippen geht immer schief! Copy + paste über den Browser klappt!); Last Will: leer lassen; „create“

4. MQTT-Abo anlegen

Die eben angelegte Verbindung anklicken, „Topic“ (= Feld mit euren Daten) abonnieren (subscribe).
Topic: AppID/devices/deviceID/up zeigt alle Daten eures devices an.
Topic: AppID/devices/deviceID/up/EUERDATENFELD zeigt nur das gewünschte Feld.
EUERDATENFELD heißt Temperatur, wenn ihr unser Beispiel nicht geändert habt.
Mit “publish” könnte man auch selbst MQTT senden.

Cooler Anwendungen? Tolle Bildungsmaterialien?

Daten in der Konsole sind toll aber mäßig sexy.

Wer mehr mit uns ausprobieren mag, ist herzlich willkommen.

Beispiele für schicke LoRaWAN-Anwendungen von Bürgern:

- Umweltsensoren: <https://skopjepulse.mk/>
- Badewasser im Dortmund-Ems-Kanal: <http://wiekaltistderkanal.de/ui/#/0>

Wer mit uns Bildungsmaterial und Unterlagen für die Hackingbox IoT entwickeln mag, ist ebenfalls herzlich willkommen.

Carolin Clausnitzer
clausnitzer@technologiestiftung-berlin.de
[@caroclausnitzer](https://twitter.com/caroclausnitzer)

Anhang

1. **Arduino-IDE / Linux / Zugriff auf serielle Schnittstelle**

Herausfinden, zu welcher group die Schnittstellenuser gehören:

`ls -l /dev/ttyUSB*` oder `ls -l /dev/ttyACM*` liefert etwas in der Art:

```
crw-rw---- 1 root tty 188, 0  5 apr 23.01 ttyUSB0
```

Die Angabe in der vierten Spalte (hier „tty“) zeigt, welche Nutzergruppe Zugriff hat.

Dieser Gruppe müssen wir noch beitreten (<username> ist unser Linux-Benutzername):

```
sudo usermod -a -G tty <username>
```

Jetzt noch mal neu starten, dann sollte der Zugriff klappen.

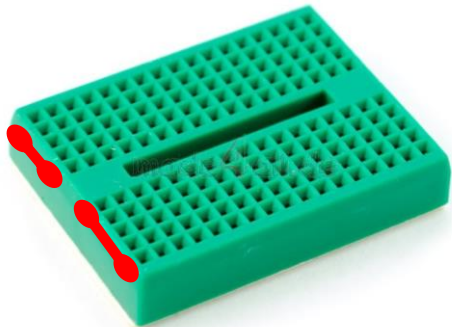
Anhang 2: optionale Kontroll-LED

Kontroll-LED: Arduino und Breadboard:

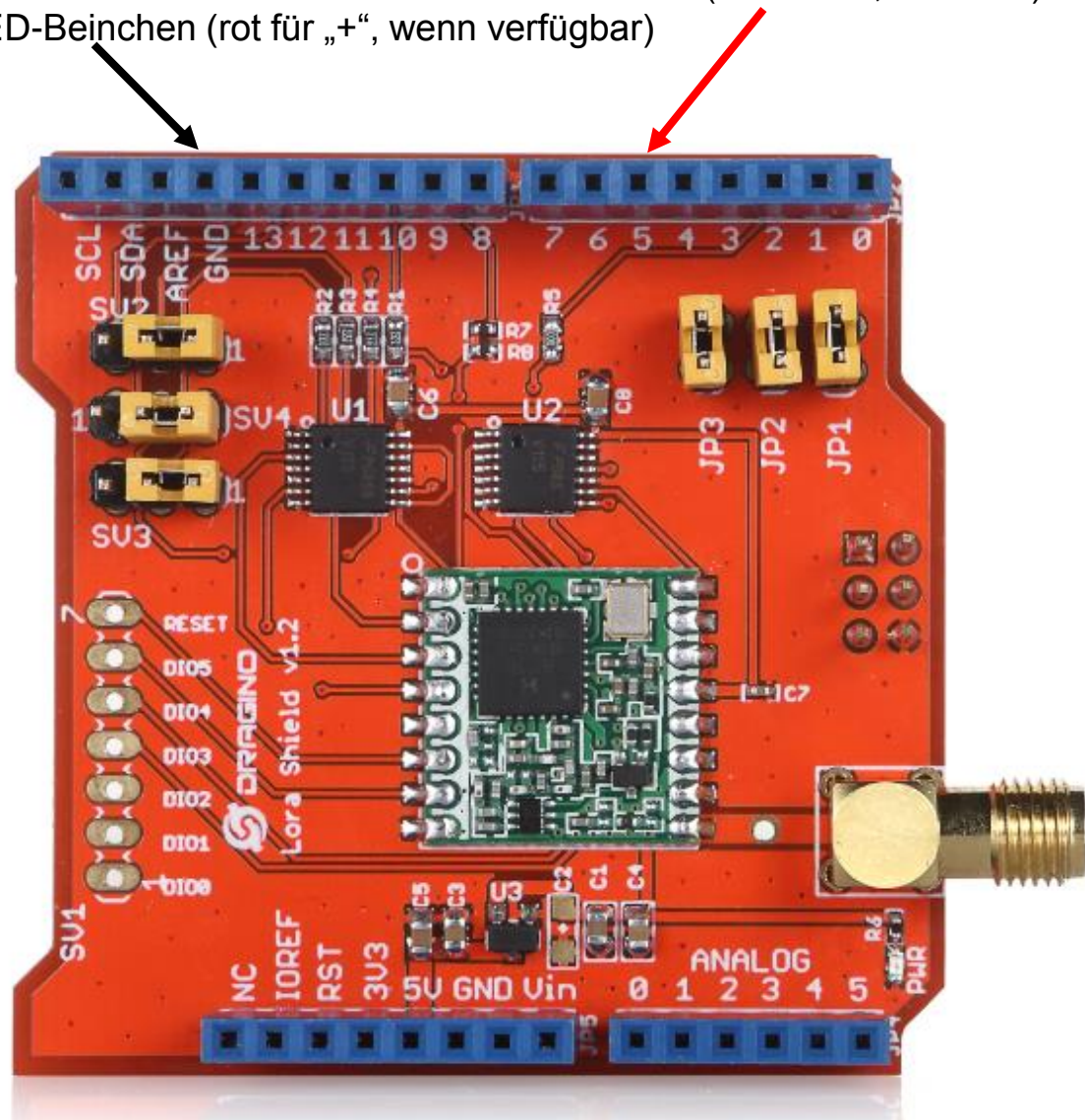
- Pin GND (auf der „Digitalseite“ vom Arduino → Vorwiderstand → kurzes LED-Bein (sw. Kabel, wenn da)
- Pin 5 (Arduino, Digitaleseite) -> langes LED-Beinchen (rot für „+“, wenn verfügbar)

Mit der LED kann man auch ohne Computer testen, ob der Sensor richtig misst.

Beim Steckbrett sind einzelne Anschlüsse miteinander verbunden



Diese 5 Anschlüsse sind verbunden.



Anhang 2: optionale LED mit Breadboard verkabeln

Kontroll-LED: Arduino und Breadboard:

- Pin GND (auf der „Digitalseite“ vom Arduino → Vorwiderstand → kurzes LED-Bein (sw. Kabel, wenn da)
- Pin 5 (Arduino, Digitale Seite) → langes LED-Beinchen (rot für „+“)

..... im Breadboard

