Kommunikation fürs IoT

Pi and More 10

Nico Maas

Zur Person

Nico Maas

IT Systemelektroniker

Bachelor of Science

mail@nico-maas.de

www.nico-maas.de

@nmaas87

Agenda

- **I. Einführung**
  - Was ist Sigfox
  - Welche Besonderheiten hat Sigfox?
  - Wie ist der Verbreitungsgrad?
- **II. Demo mit Wisol EVK**
  - Evaluation Kit besorgen (hier: Wisol EVK EVBSFM10R Rev.3)
  - Evaluation Kit registrieren
  - Callback erstellen
  - Nachrichten senden
- **III. Abschluss**
  - Fragen
  - Quellenangaben

Full Disclosure

Dieser Vortrag wurde in keinem Bereich von Sigfox finanziell unterstützt oder gesponsert.

Allerdings wurden von Wisol Evaluation Kits bereitgestellt um echte Erfahrung mit dem Sigfox Netzwerk und der Wisol Hardware zu erlangen.

Vielen Dank an Wisol!

I. Einführung

Einführung: Sigfox

- **Sigfox:**
  - Baut keine Chips oder fertige Produkte
  - Design vom Protokoll, Betrieb vom Carrier Network und Lizenzierung
- **Prinzipien:**
  - Low Power Wide Area Network
  - Low Cost
  - Low energy consumption
  - Ease of use
  - Long range
  - Operated ("Carrier Network")
  - Frequency-independent (ISM Band)
  - Embedded subscriber identification (DEVID + PAC [Porting Authorization Code])
  - Penetration (Structures / Anti-Jam -> Ultra Narrow Band)

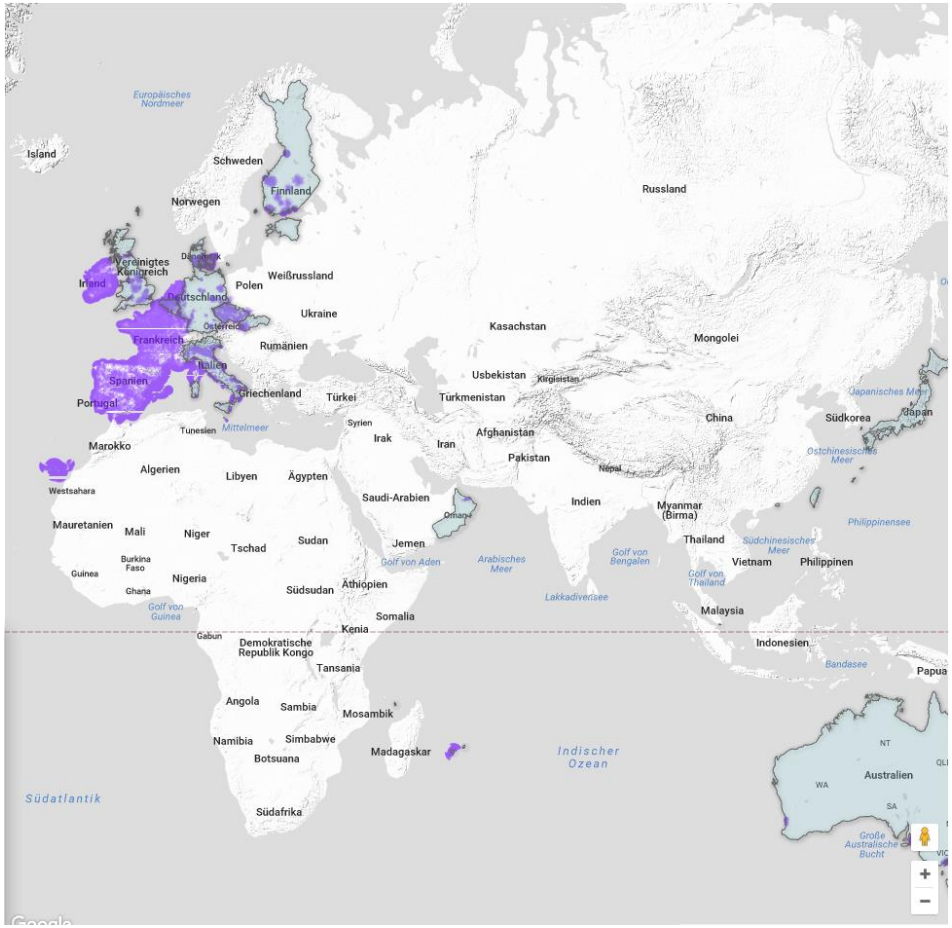
Einführung: Sigfox

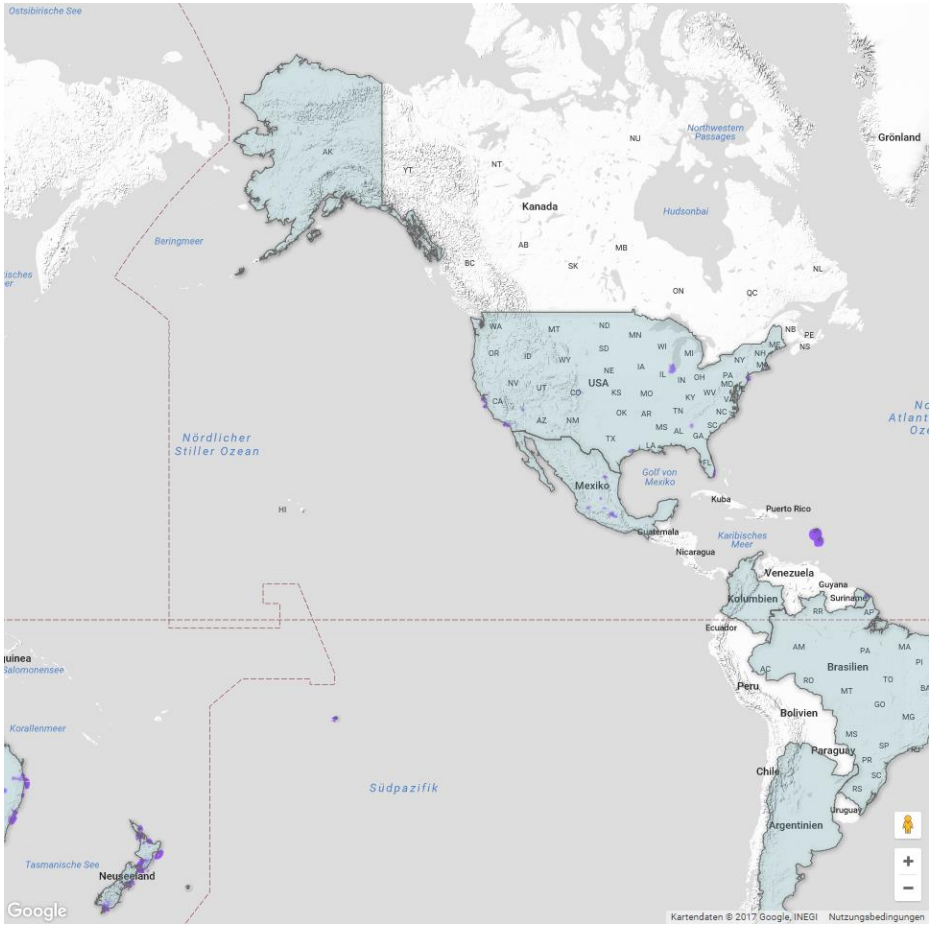
- **ISM Band (868 MHz Europa / 915 MHz USA)**
  - unterschiedliche Zonen:
    - RCZ1 Europe
    - RCZ2 US
    - RCZ3 TBD
    - RCZ4 Australia/New Zealand
- **Reichweite: 30 – 50 km (ländliche Gegend), 3 – 10 km (Stadt)**
- **Max. Reichweite: +100 km bei LoS**
- **Ca. 1 Millionen "Things" pro Station**
- **Energiebedarf pro Station 1/1000 im Vergleich zu GSM**
- **Tx: <50mA, für wenige Sekunden, 25mW, 14dB / Standby: ~ µA**

Einführung: Sigfox

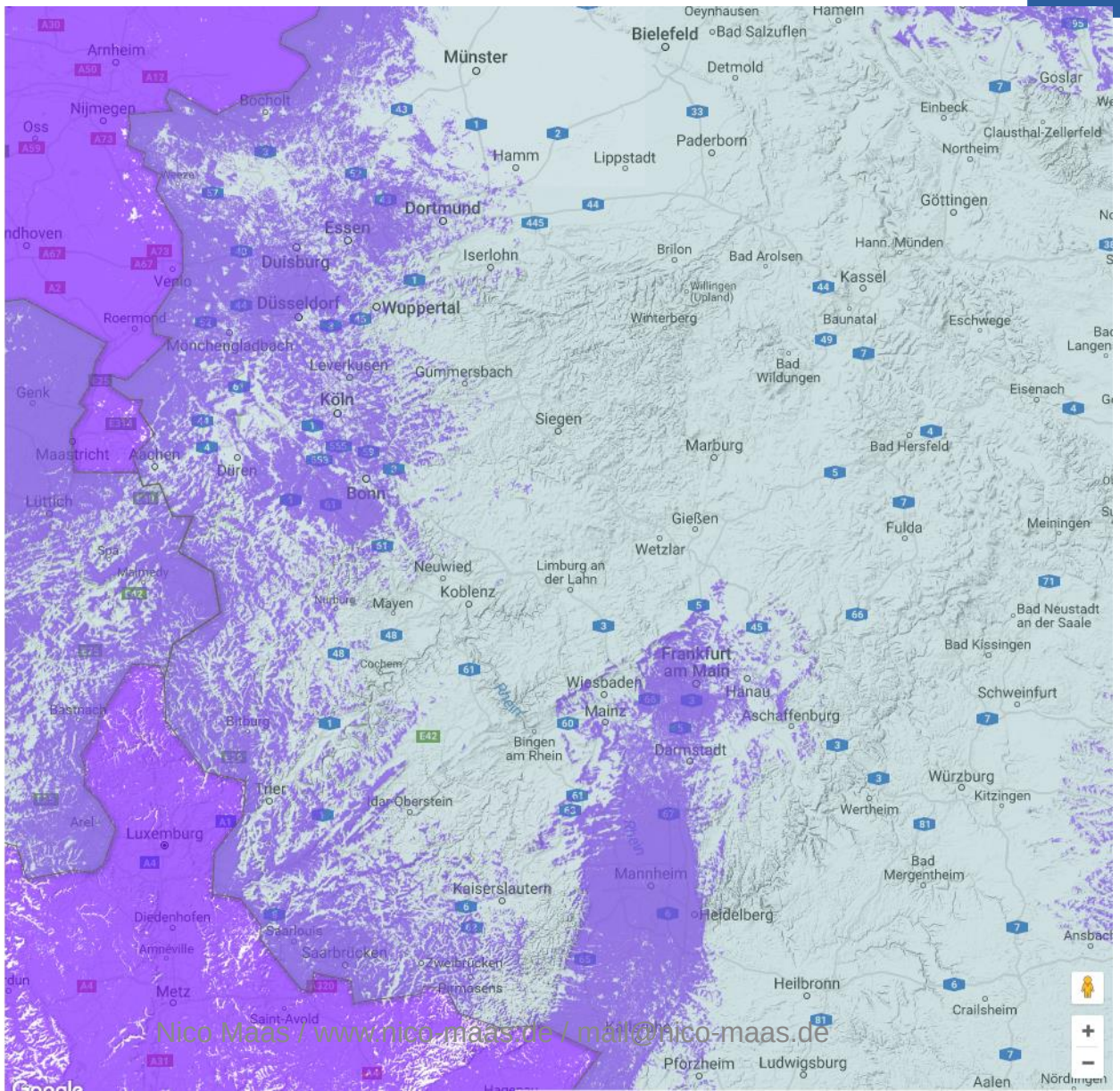
- **Bidirektional**
- **Durchsatz: 100 bits / sec**
- **Uplink**
  - Payload: 12 Bytes = 96 Bit
  - Nachrichten: 140 / Tag
- **Downlink**
  - Payload: 8 Bytes = 64 Bit
  - Nachrichten: bis zu 4 / Tag
  - "Due to ETSI regulation, [...] devices are limited to 4 [...] messages a day." (http://makers.sigfox.com/getting-started/)
- **Jede Kommunikation wird vom Client initialisiert**
- **RESTful API (JSON Payloads)**

Einführung: Verbreitung





Einführung: Verbreitung



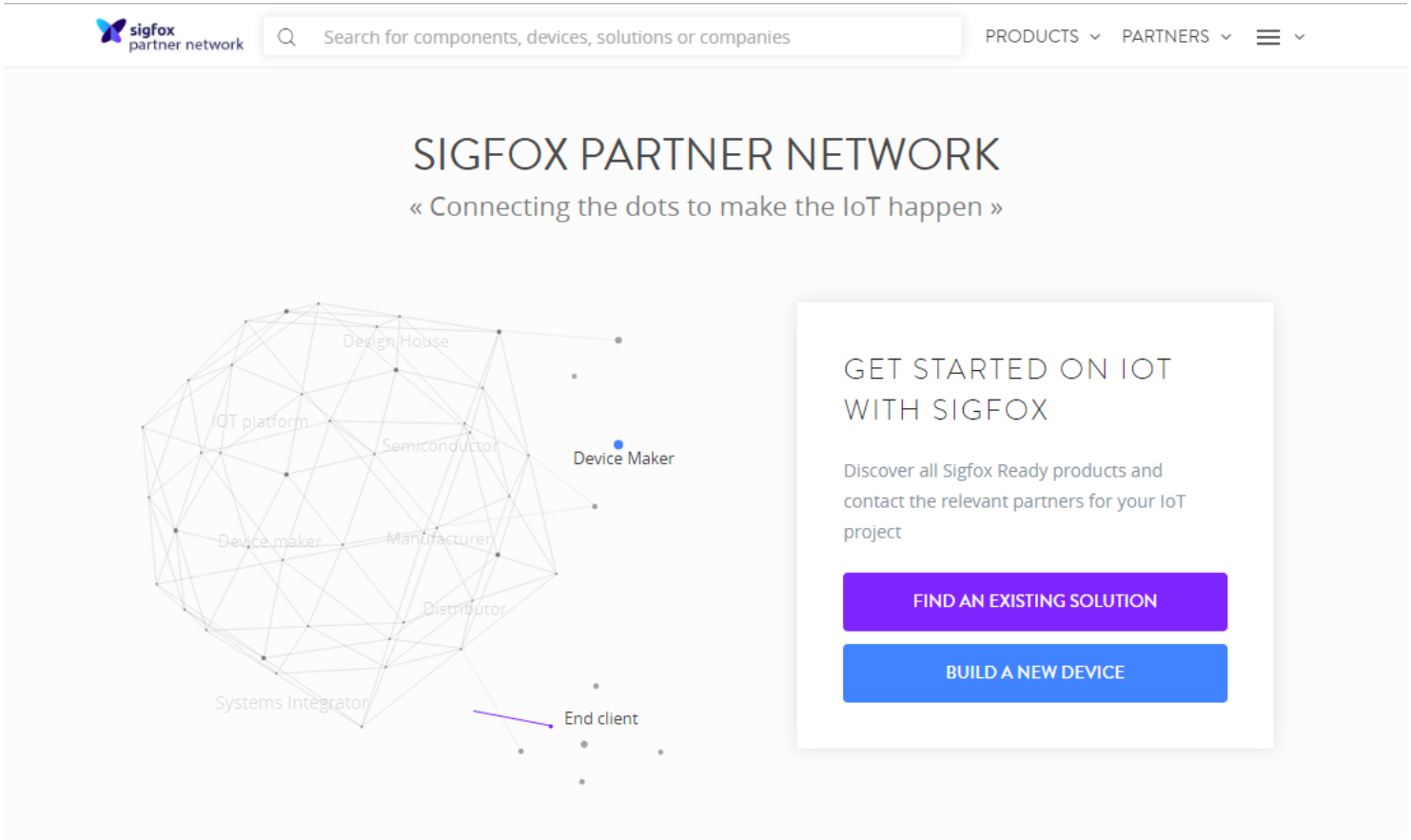
II. Demo mit Wisol EVK

Demo

- **I. Evaluation Kit besorgen**
- **II. Evaluation Kit registrieren**
- **III. Callback erstellen**
- **IV. Nachrichten senden**

Demo: I. Evaluation Kit besorgen

- **Partner System anmelden:**
  - https://partners.sigfox.com/

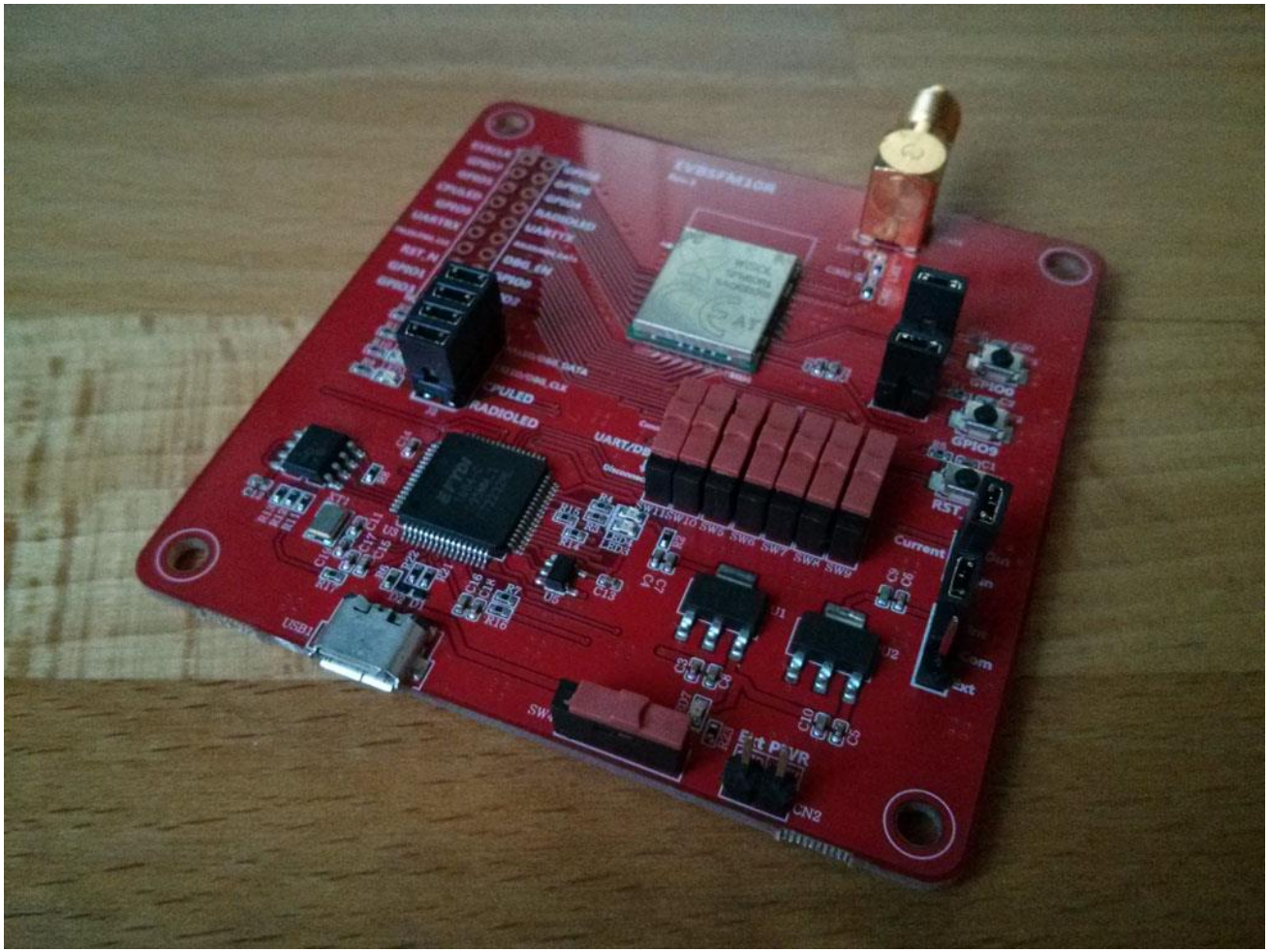


Demo: I. Evaluation Kit besorgen

- **Richtigen Supplier finden + Evalkit für die RCZ1 Zone (Europa)**
- **ggf. bei deren Website anmelden**
- **Lieferbedingungen klären und bestellen**
- **Spaß mit Zoll und Co KG (Einführzölle und Liefergebühren betrachten!)**

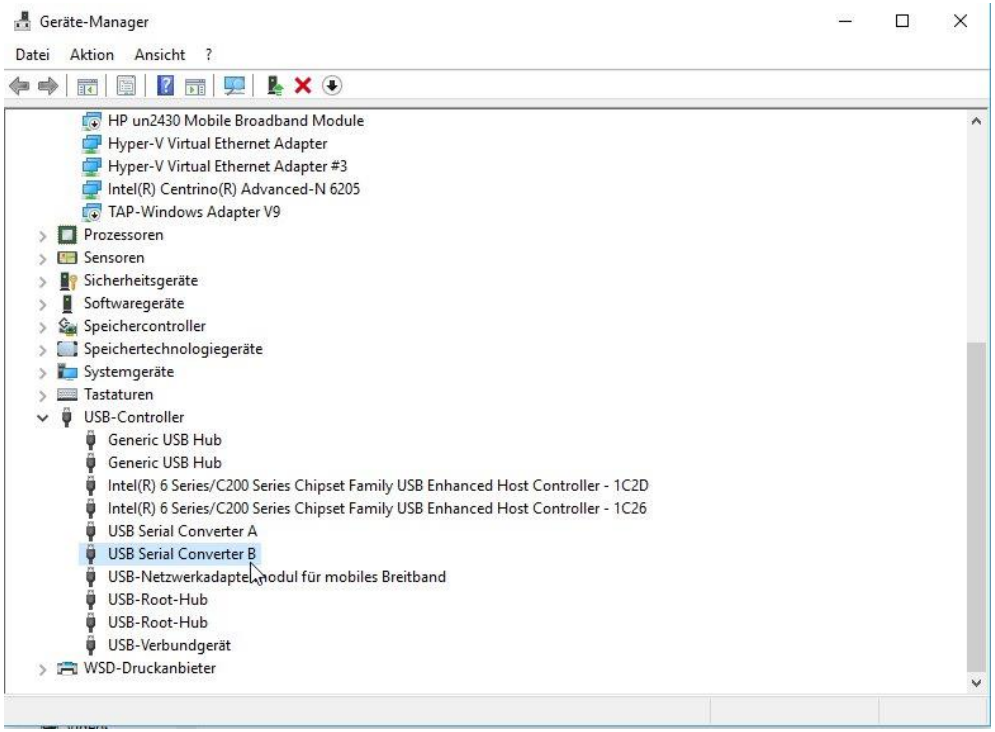
Demo: I. Evaluation Kit besorgen

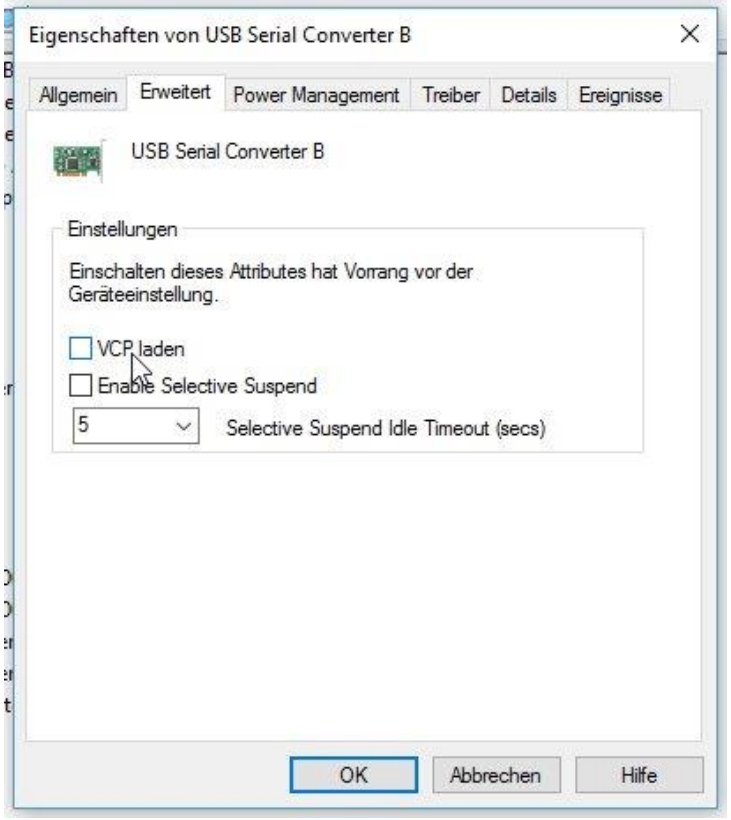
- **Wisol EVBSFM10R1**



Demo: II. Evaluation Kit registrieren

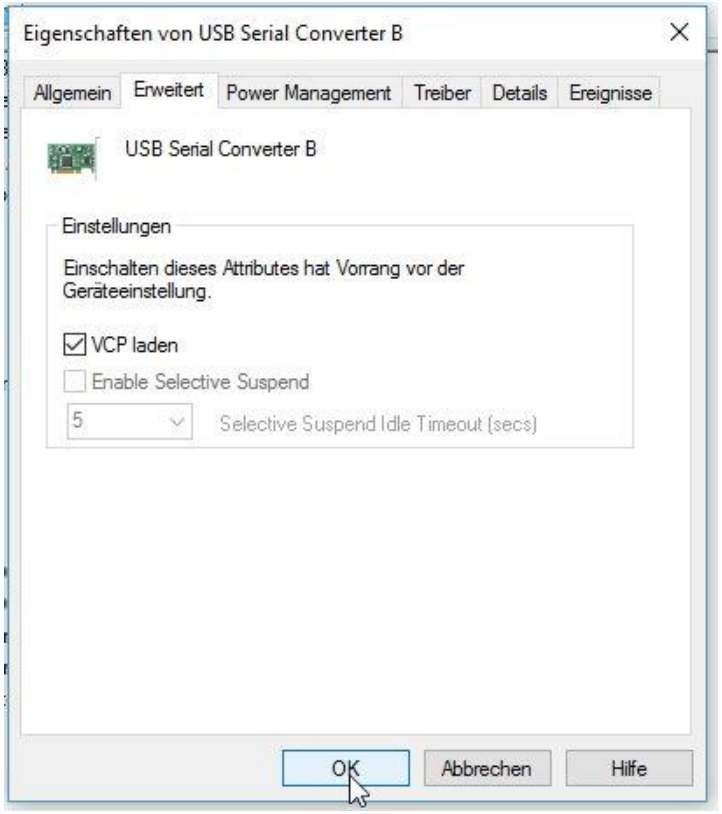
- **Treiber herunterladen und installieren**
- **VCP aktivieren**

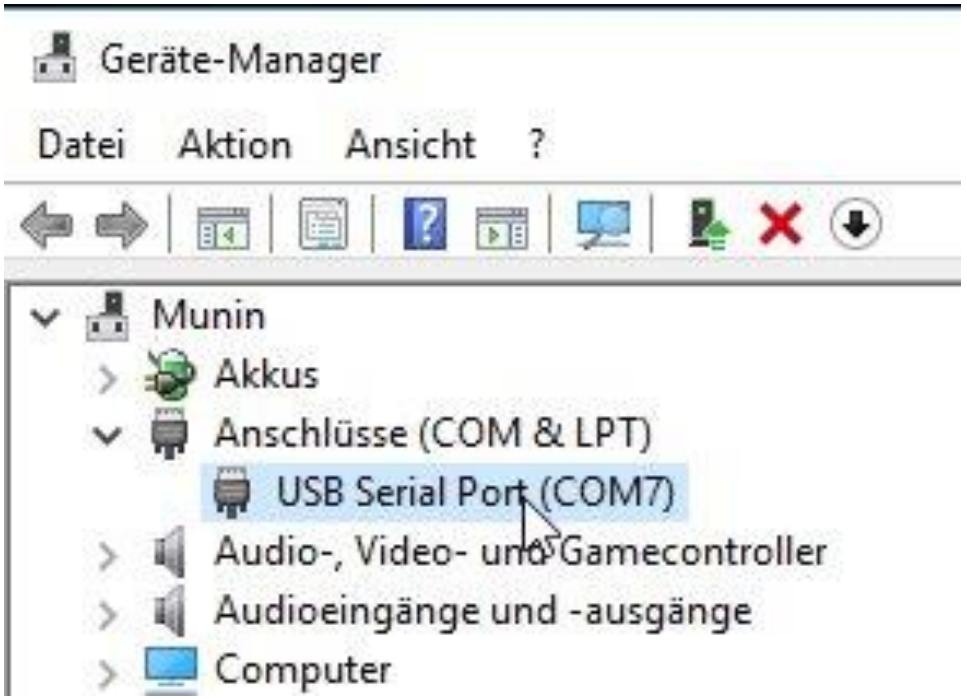




Demo: II. Evaluation Kit registrieren

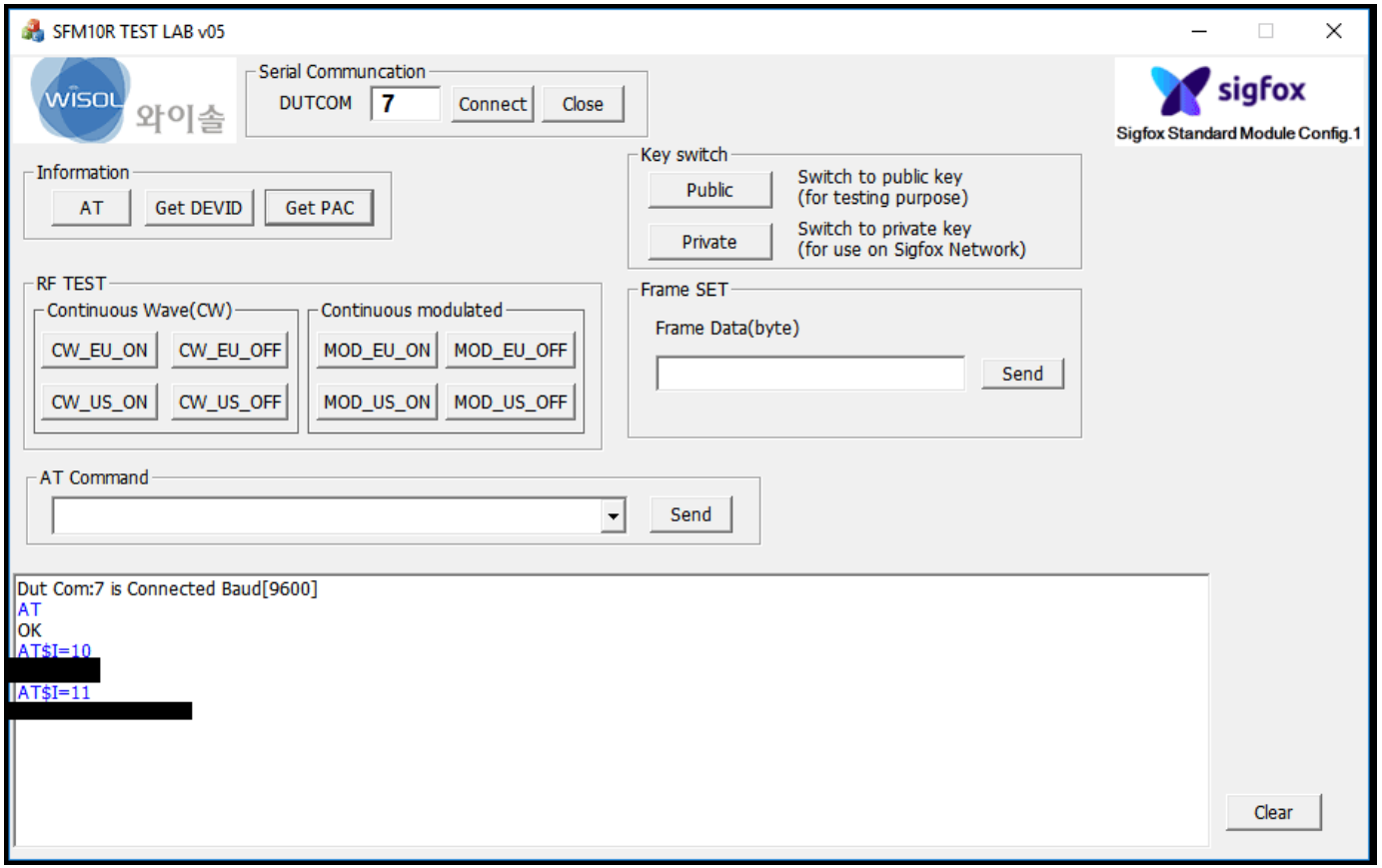
- **Evalboard nochmal abstecken und neu anschließen**
- **Gerät erscheint als Serial Anschluss**





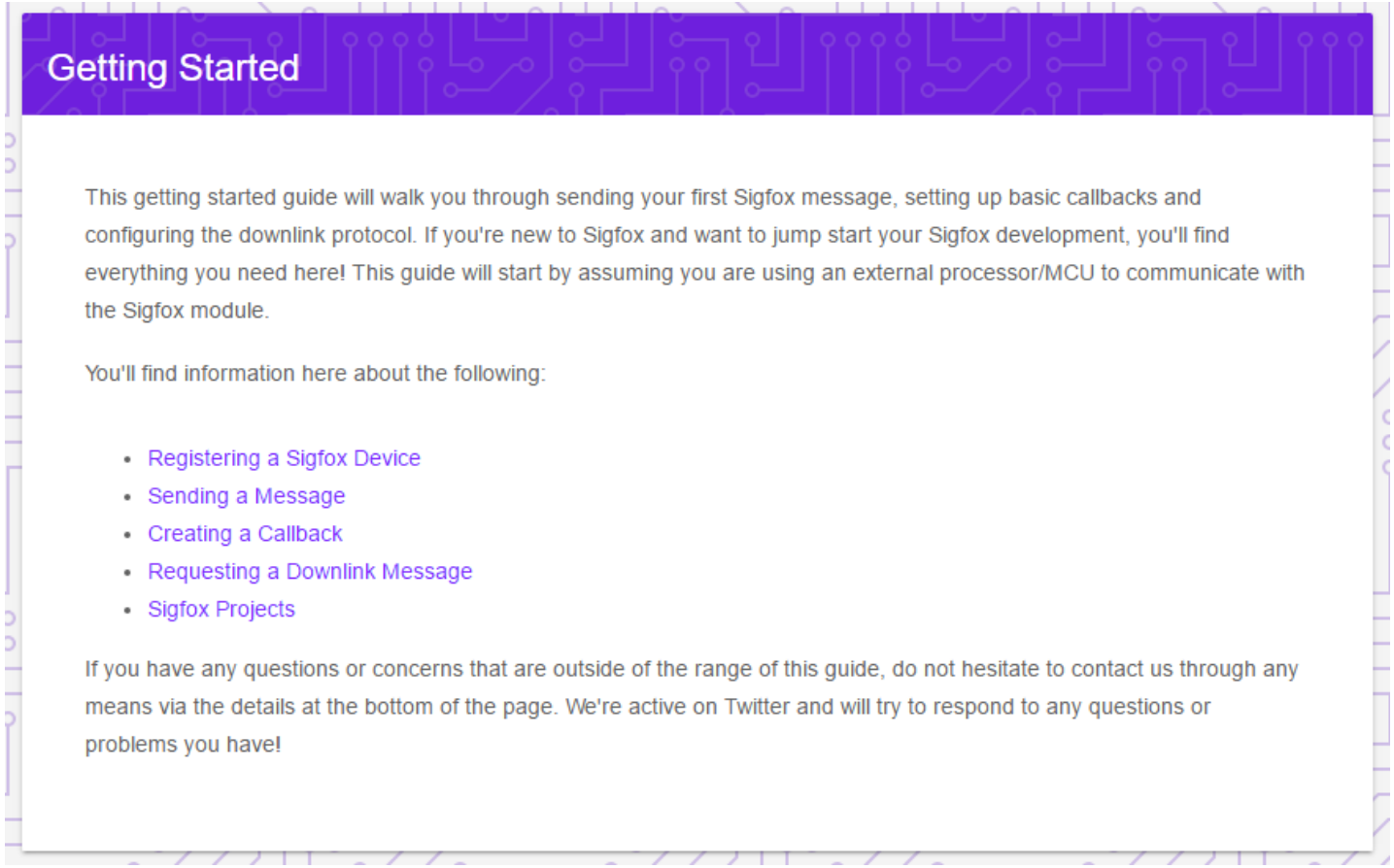
Demo: II. Evaluation Kit registrieren

- **Testsoftware herunterladen und starten**
- **COM Nummer eingetragen, verbinden und DEVID und PAC auslesen**



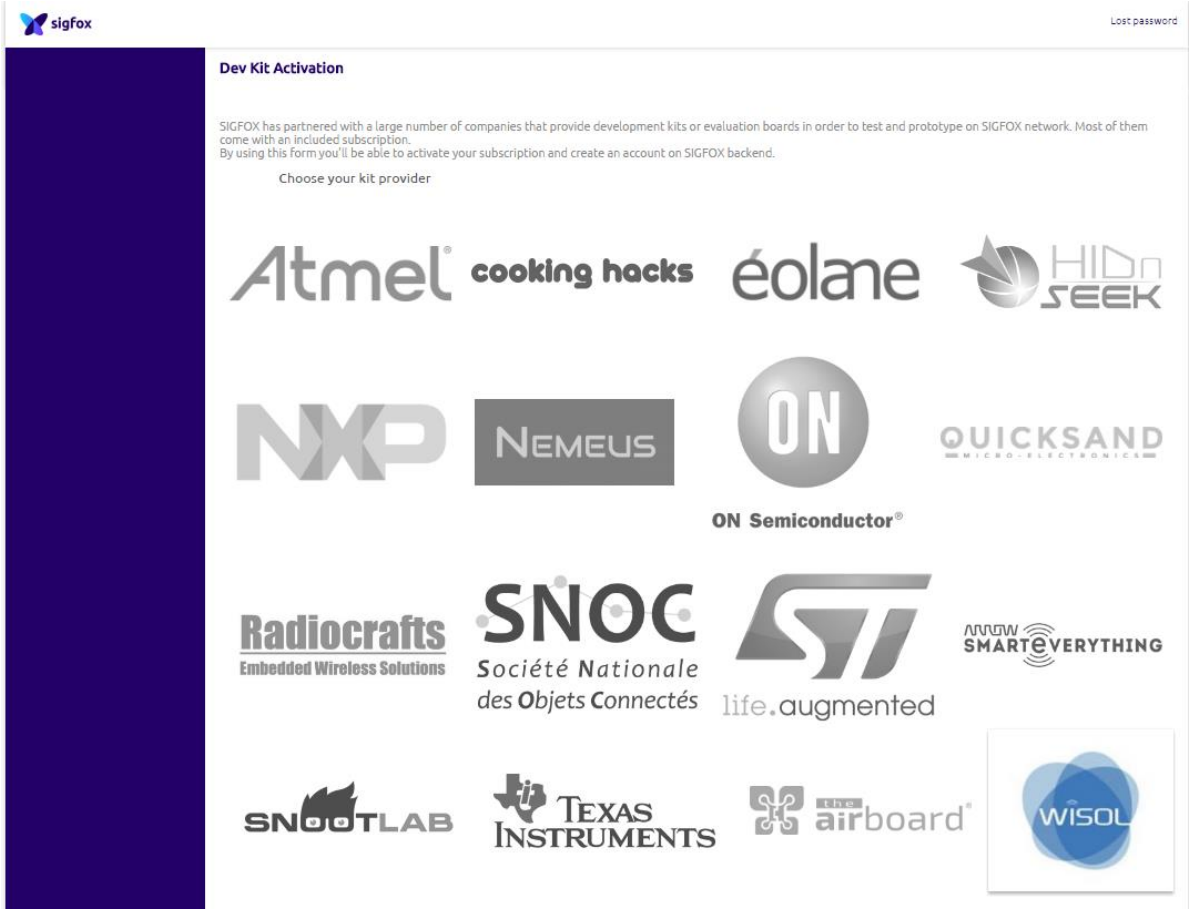
Demo: II. Evaluation Kit registrieren

- **Auf "Getting Started" Website gehen:**
  - http://makers.sigfox.com/getting-started/



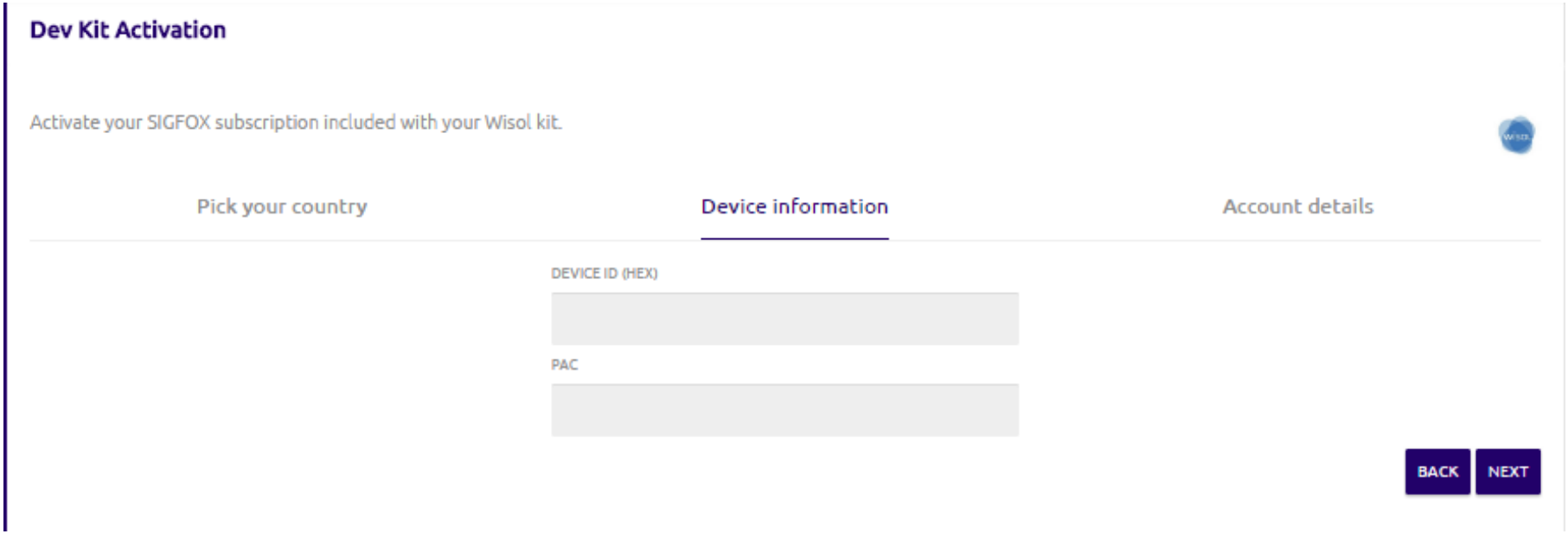
Demo: II. Evaluation Kit registrieren

- **Registrierung**
  - https://backend.sigfox.com/activate



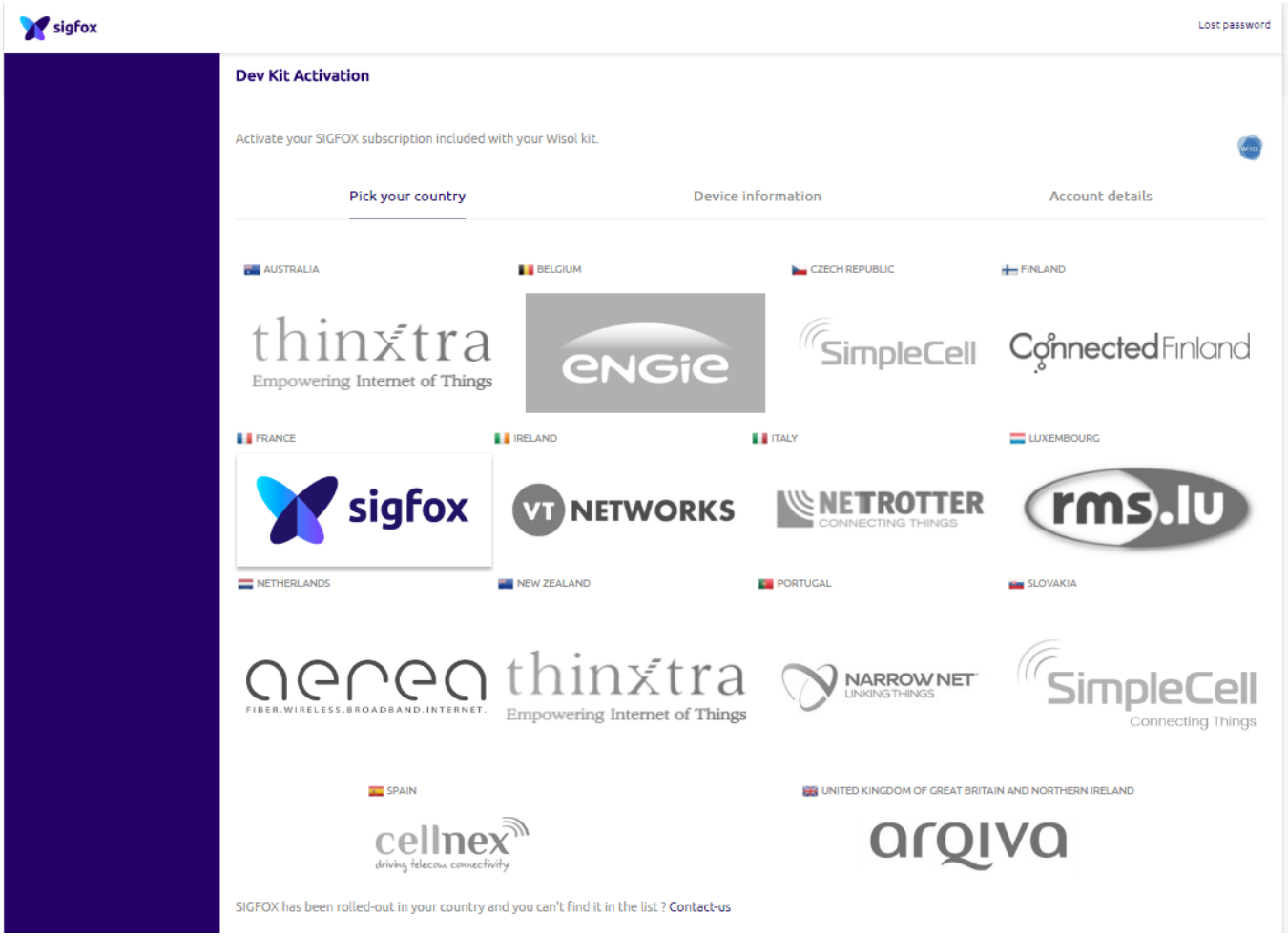
Demo: II. Evaluation Kit registrieren

- **DEVID und PAC eingeben**



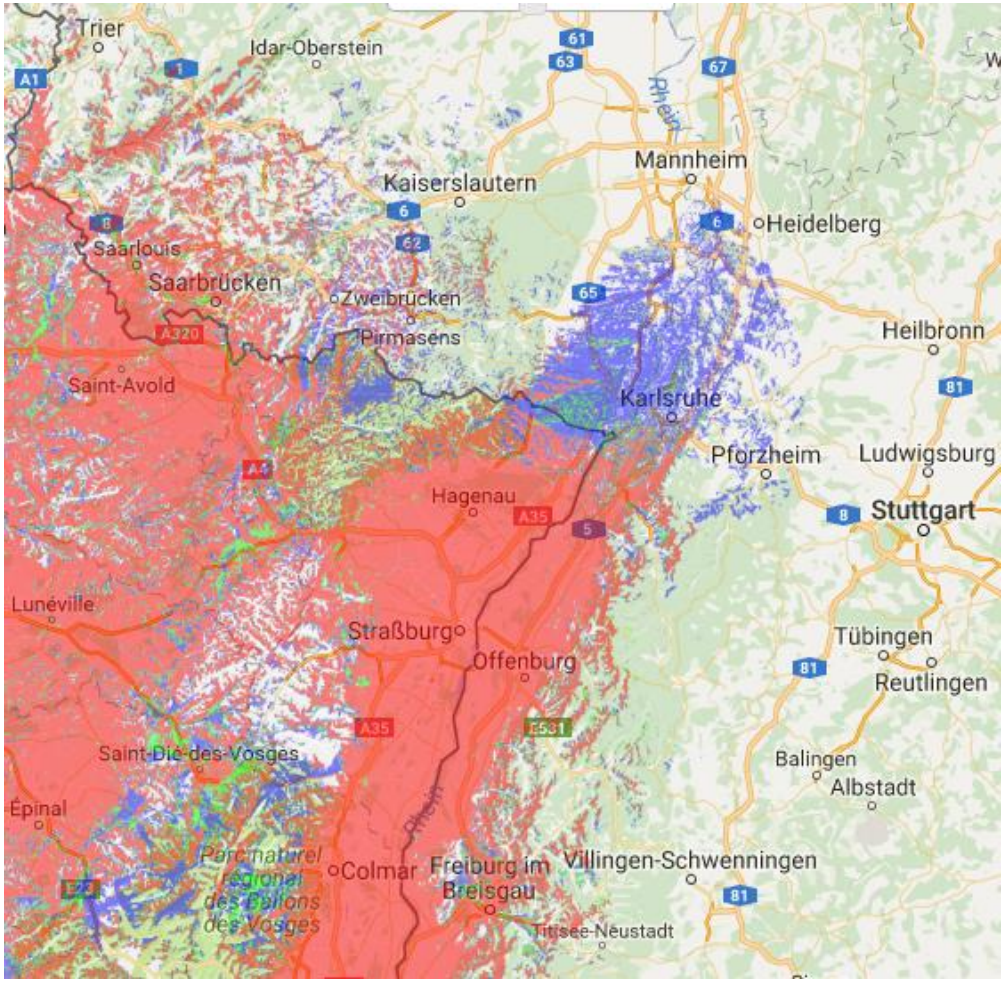
Demo: II. Evaluation Kit registrieren

- **Provider auswählen**



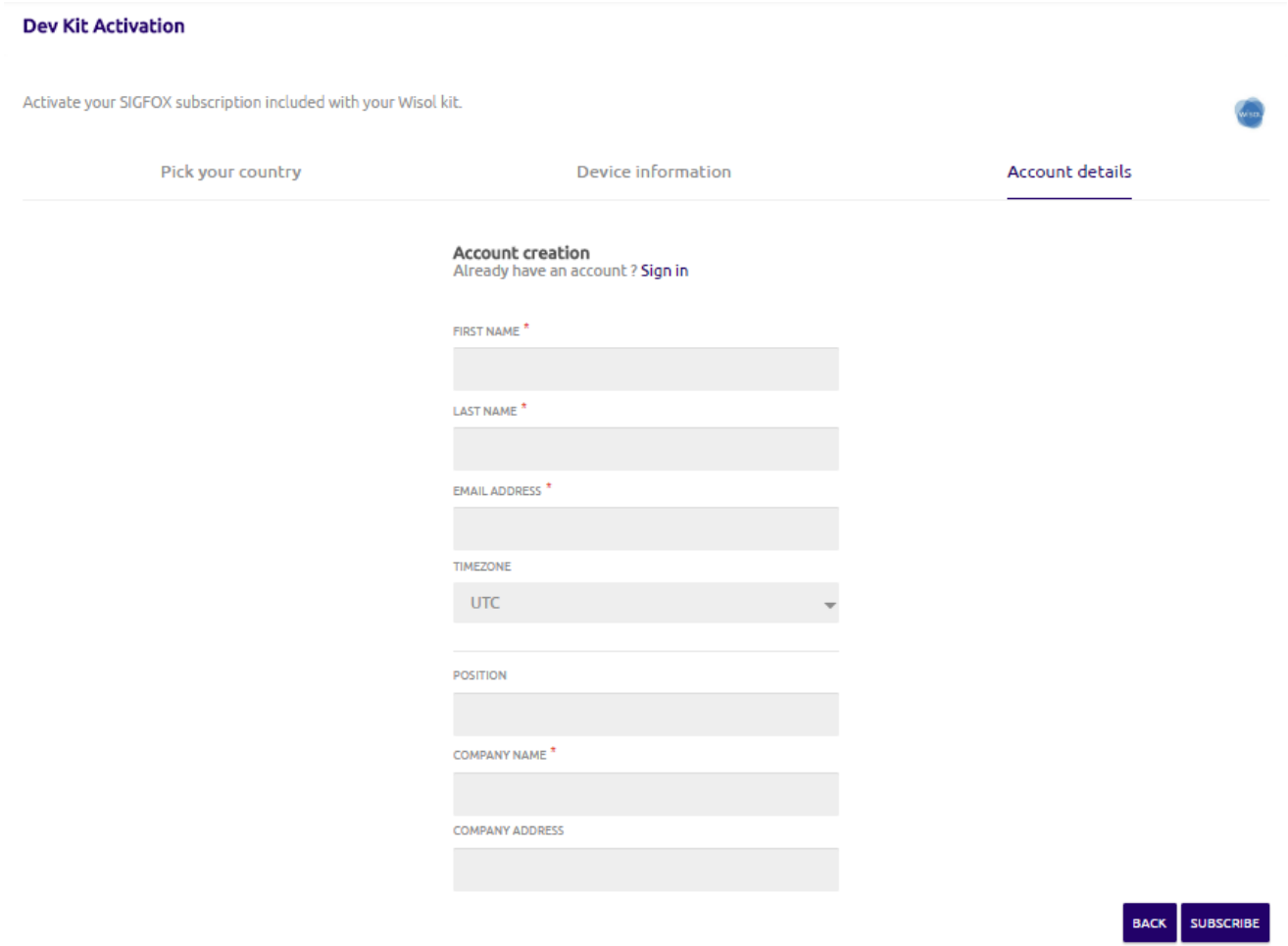
Demo: II. Evaluation Kit registrieren

- **Provider auswählen**



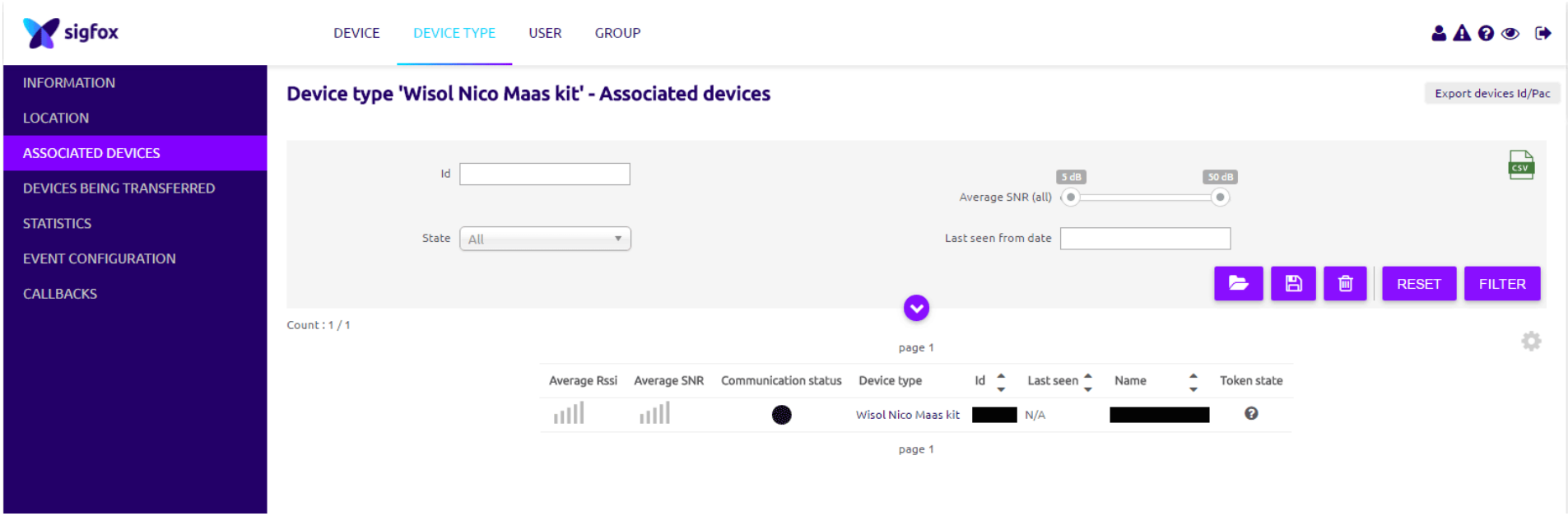
Demo: II. Evaluation Kit registrieren

- **Account anlegen**



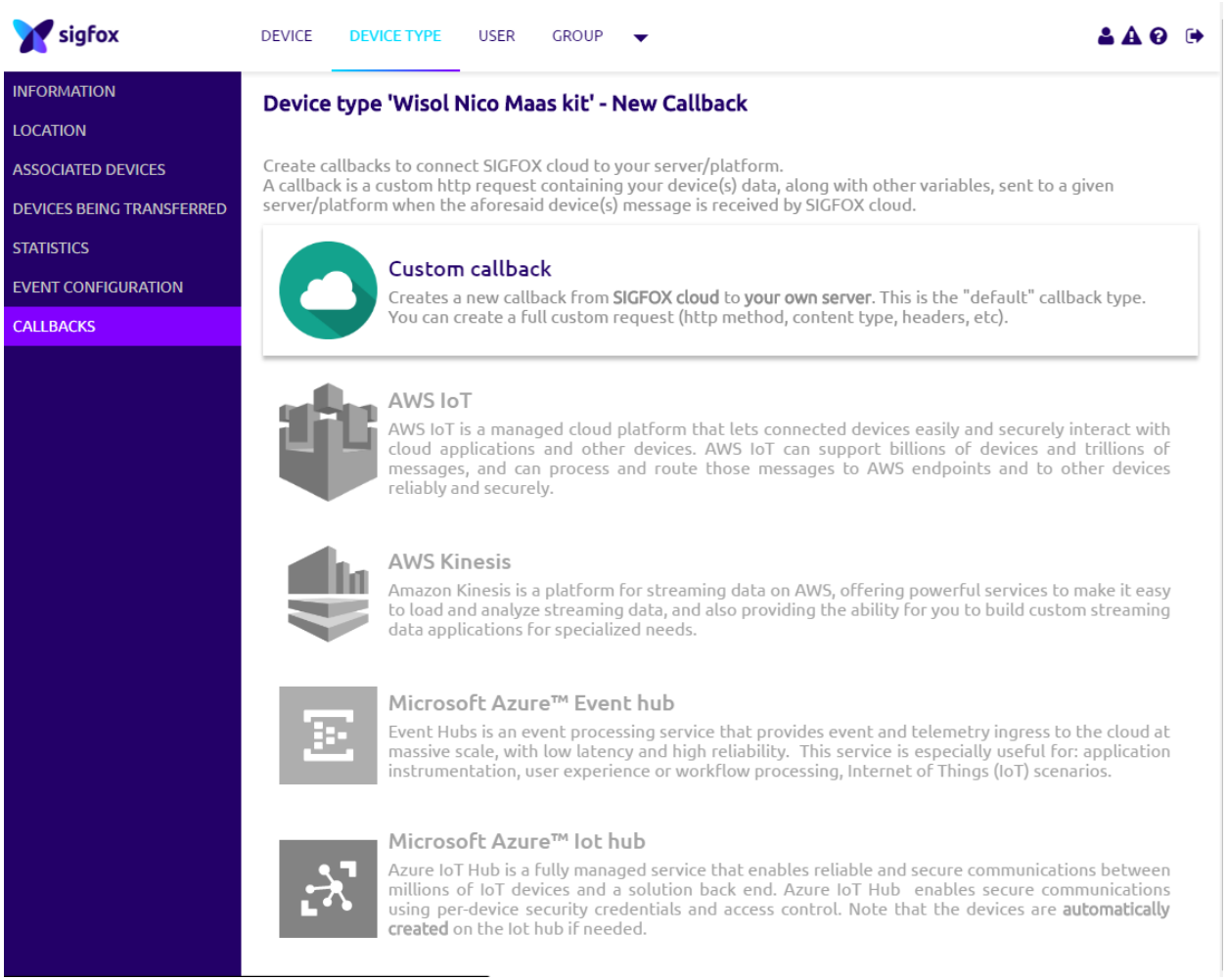
Demo: II. Evaluation Kit registrieren

- **Übersicht**



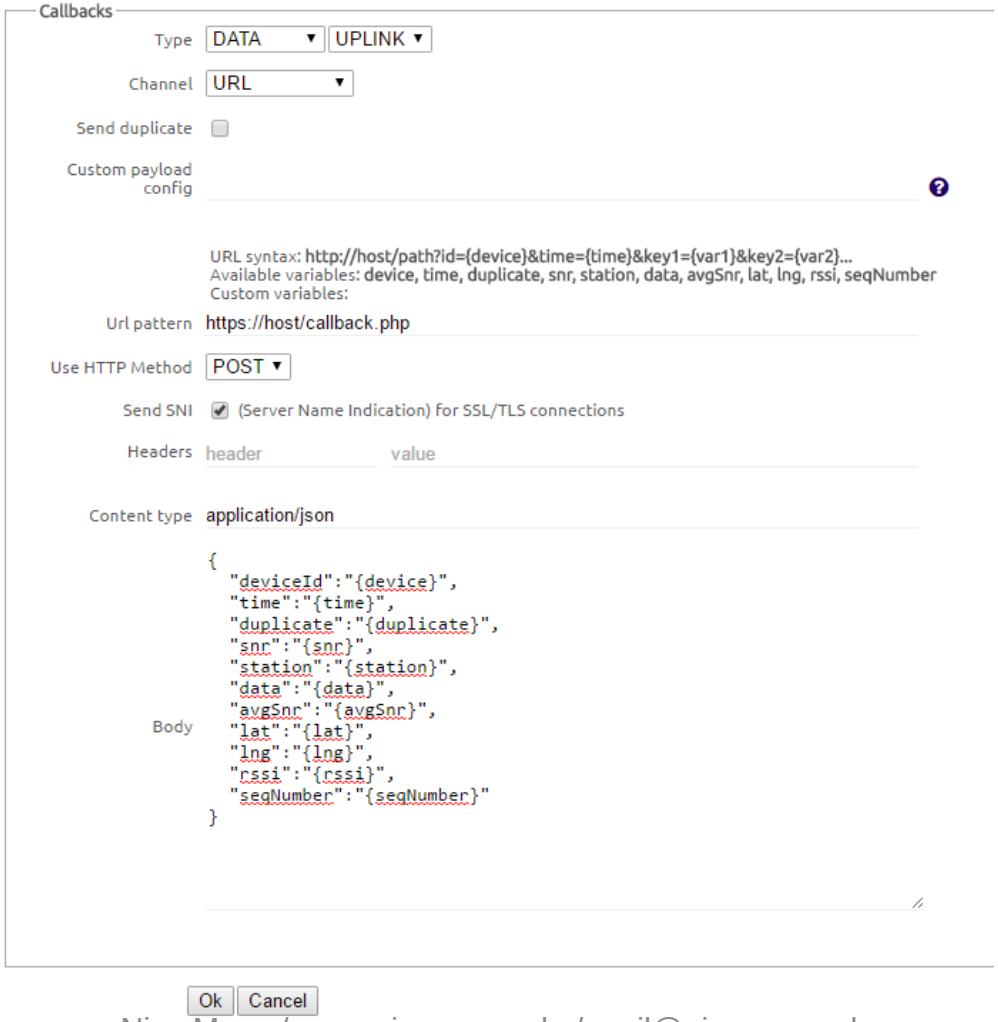
Demo: III. Callback erstellen

- **Auf Device Type klicken, Callbacks, New**



Demo: III. Callback erstellen

- **Callback definieren (https://backend.sigfox.com/apidocs/callback)**



Demo: III. Callback erstellen

- **Ggf. Custom payload definieren**

Custom payload config

Custom message type decoding grammar

The "custom format" grammar is as follows :

```
format = field_def [" " field_def]* ;
field_def = field_name ":" byte_index ":" type_def ;
field_name = (alpha | digit | "#" | "_")* ;
byte_index = [digit*] ;
type_def = bool_def | char_def | float_def | uint_def ;
bool_def = "bool:" ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7") ;
char_def = "char:" length ;
float_def = "float:" ("32" | "64") [ ":little-endian" | ":big-endian" ] ;
uint_def = "uint:" ("8" | "16" | "24" | "32") [ ":little-endian" | ":big-endian" ] ;
int_def = "int:" ("8" | "16" | "24" | "32" | "40" | "48" | "56" | "64") [ ":little-endian" | ":big-endian" ] ;
length = number* ;
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

A field is defined by its name, its position in the message bytes, its length and its type :

- the field name is an identifier including letters, digits and the '-' and '_' characters.
- the byte index is the offset in the message buffer where the field is to be read from, starting at zero. If omitted, the position used is the current byte for boolean fields and the next byte for all other types. For the first field, an omitted position means zero (start of the message buffer)

Next comes the type name and parameters, which varies depending on the type :

- **boolean** : parameter is the bit position in the target byte
- **char** : parameter is the number of bytes to gather in a string
- **float** : parameters are the length in bits of the value, which can be either 32 or 64 bits, and optionally the endianness for multi-bytes floats. Default is big endian. Decoding is done according to the IEEE 754 standard.
- **uint** (unsigned integer) : parameters are the number of bits to include in the value, and optionally the endianness for multi-bytes integers. Default is big endian.
- **int** (signed integer) : parameters are the number of bits to include in the value, and optionally the endianness for multi-bytes integers. Default is big endian.

Examples :

Format	Message (in hex)	Result
int1::uint:8 int2::uint:8	1234	{ int1: 0x12, int2: 0x34 }
b1::bool:7 b2::bool:6 i1:1:uint:16	C01234	{ b1: true, b2: true, i1: 0x1234 }
b1::bool:7 b2::bool:6 i1:1:uint:16:little-endian	801234	{ b1: true, b2: false, i1: 0x3412 }
b1::bool:7 b2::bool:6 i1:1:uint:16:little-endian i2::uint:8	80123456	{ b1: true, b2: false, i1: 0x3412, i2:0x56 }
str::char:6 i1::uint:16 i2::uint:32	414243444546012345678 90A	{ str: "ABCDEF", i1: 0x123, i2:0x4567890A }

Demo: III. Callback erstellen

- **Callback auf Server programmieren**

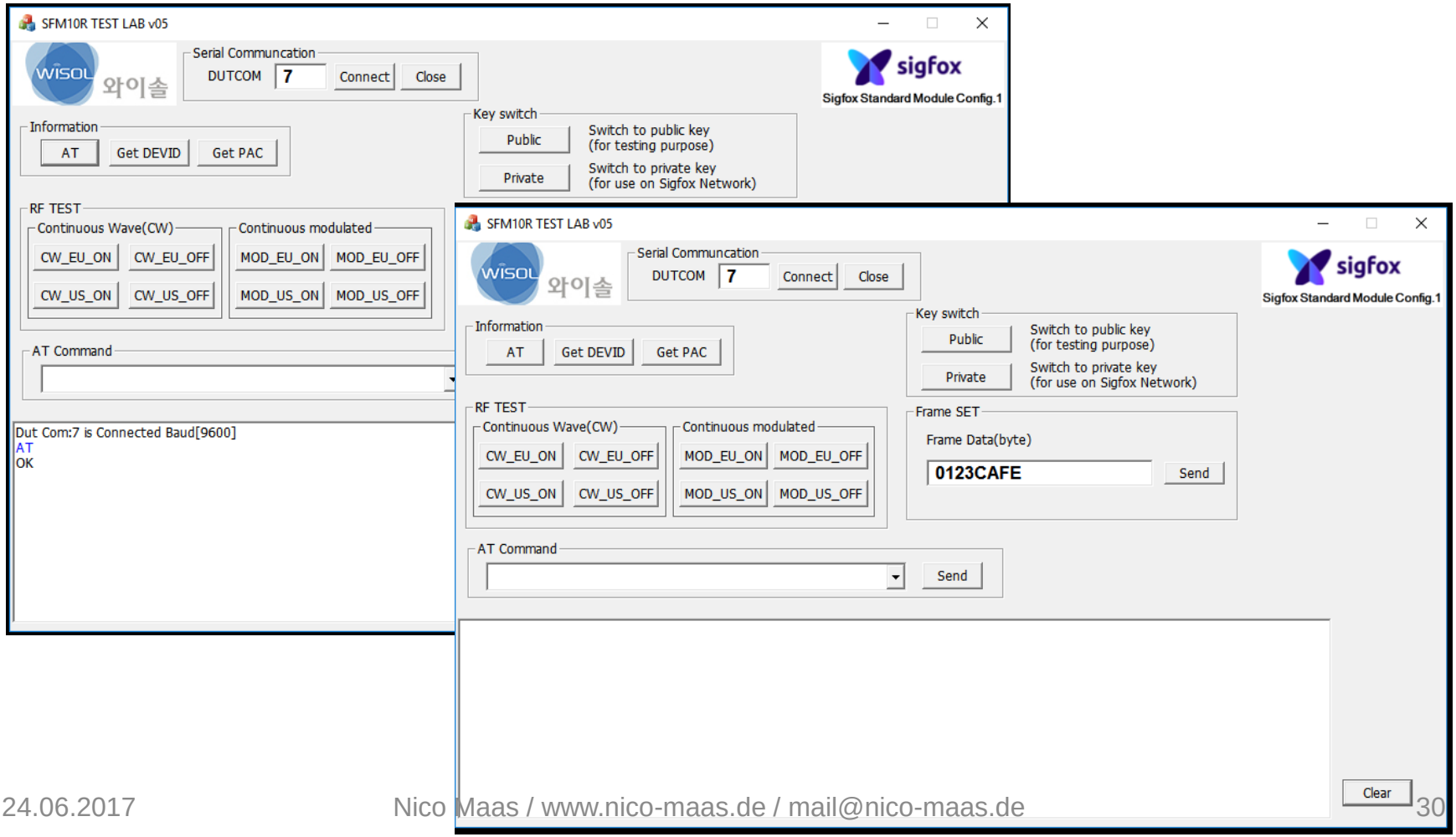
```
<?php
header('Content-type: application/json');
$json = file_get_contents('php://input');
$arr = json_decode($json,true);

function writeFile_Sigfox($write){
$fh = fopen('sigfox.txt', 'a') or die("can't open file");
fwrite($fh, $write);
fclose($fh);
}

writeFile_Sigfox($json);
writeFile_Sigfox($arr['deviceId']);
?>
```

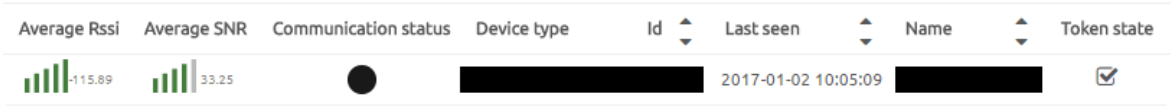
Demo: IV. Nachrichten senden

- **Testsoftware starten, verbinden, Private Key auswählen, Nachricht eingeben und senden**



Demo: IV. Nachrichten senden

- **Nachricht wird in Callback empfangen**




```
{
  "deviceId":"xxxxxx",
  "time":"1483395345",
  "duplicate":"false",
  "snr":"9.63",
  "station":"xxxx",
  "data":"0123cafe",
  "avgSnr":"23.91",
  "lat":"50.0",
  "lng":"6.0",
  "rssi":"-137.00",
  "seqNumber":"82"
}
```


15,93 km

III. Abschluss

Abschluss: Zusammenfassung

- **Vorteile Sigfox:**
  - ideal für kleine Sensoren
  - verbraucht wenig Strom -> lange Standby Zeiten

- **Nachteile Sigfox:**
  - Keine Bestätigung des Nachrichten Empfanges
  - Nur beschränkte Möglichkeit zum Sensor selbst zu kommunizieren (4 Nachrichten pro Tag)
  - Kosten / Subscription relativ undurchsichtig (zw. 14€ und 1€ pro Gerät / Jahr, ohne Gewähr)
  - Security?

Fragen?

Vielen Dank für Ihre Aufmerksamkeit!

Quellenangaben

- Bilder, soweit nicht anders angegeben, von sigfox.com oder selbst erstellt
- http://www.eejournal.com/archives/articles/20141103-sigfox/
- http://www.rfwireless-world.com/Terminology/SIGFOX-technology-basics.html
- https://github.com/sigfox/makers-tour/blob/master/presentation/2016/presentation.pdf
- https://radiocrafts.com/uploads/AN018_SIGFOX_Frequenty_Asked_Questions_2_1.pdf
- https://developers.thethings.io/serialization-formats-sigfox.html
- http://makers.sigfox.com/getting-started/