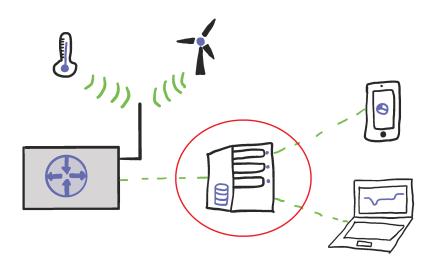




LoRa-Workshop 2 – LoRa und TheThingsNetwork Alexander Knaak, Benno Hölz SFZ Südwürttemberg

## The Things Network (TTN) Ein Server für unsere Messwerte



## The Things Network (TTN) Ein Server für unsere Messwerte

#### **Application**

- Sammelpunkt aller Geräte und Daten
- Ausgangspunkt für die Verarbeitung der Daten

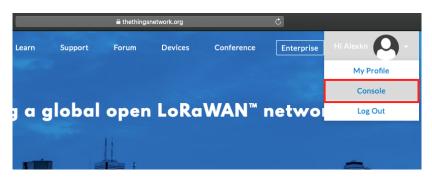
#### Device

- Sensorknoten
- Senden die Daten über das Gateway an die Application

### The Things Network (TTN) Ein Server für unsere Messwerte

- 1. Konto erstellen: https://www.thethingsnetwork.org/
- 2. Application erstellen (Console)
- 3. Device zur Application hinzufügen

Im TheThingsNetwork die Console öffnen.



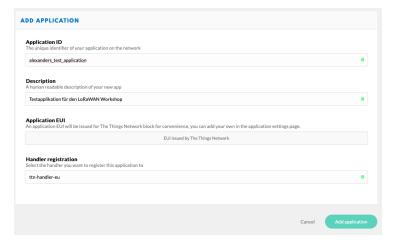
### Auf das Symbol Applications klicken



Auf add application klicken, um eine neue Applikation zu erstellen.



Es öffnet sich ein Fenster in dem Details zur Applikation eingetragen werden müssen.



Im Reiter *Payload Formats* wählen wir *Custom* und fügen folgenden Code unter *decoder* ein (mehr dazu später):

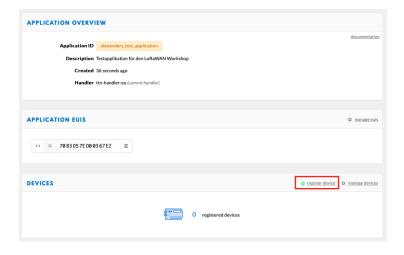
→ Bestätigen mit save payload functions!

### The Things Network (TTN) Einfügen eines Gerätes

In der erstellten Applikation können nun die einzelnen Sensorknoten (devices) angelegt werden.

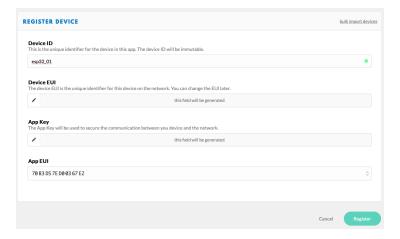
Dazu klickt man im Abschnitt DEVICES auf register device

### The Things Network (TTN) Einfügen eines Gerätes



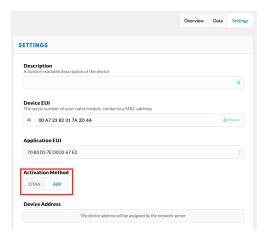
#### TheThingsNetwork (TTN) Einfügen eines Gerätes

Es öffnet sich ein neues Fenster in dem die *Device ID* des Sensors festgelegt werden muss.



## The Things Network (TTN) Einfügen eines Gerätes

In den *Settings* des Gerätes muss in unserem Fall die *Activation Method* von OTAA auf ABP geändert werden.



## The Things Network (TTN) Einfügen eines Gerätes

In der *Device Overview* können nun die wichtigsten Parameter zur Einbindung des Sensors in die LoRaWAN Applikation ermittelt werden.

EVICE OVERVIEW	
Application ID	alexanders_test_application
Device ID	esp32_01
Activation Method	ABP
Device EUI	↔ = 000 A7 23 82 31 7A 2D 4A (E)
Application EUI	↔ = 70 B3 D5 7E D0 03 67 E2 (E)
Device Address	○ = 26011558 個
Network Session Key	O = Ø msb t
App Session Key	

#### ESP und LoRaWAN Werte ins Netzwerk schicken

#### Benötigt auf dem ESP:

- boot.py (Hauptprogramm)
- config.py
- encryption\_aes.py
- ▶ sx127x.py
- ▶ ttn
  - ttn\_eu.py

### ESP und LoRaWAN Konfiguration des ESP

#### Die config.py enthält folgende Informationen:

- ▶ Pinbelegung des LoRa Chips für verschiedene Microcontroller
- Das Sendeintervall der LoRaWAN-Botschaft
- Die LoRa-Parameter (Frequenz, Kanal, Bandbreite, ...)
- Die TTN-Parameter der Applikation

### ESP und LoRaWAN Konfiguration des ESP

Da hier der ESP32 genutzt wird, muss die entsprechende Definition einprogrammiert werden. Die anderen Gerätekonfigurationen können entweder auskommentiert oder gelöscht werden.

```
1 # ES32 TTGO v1.0
2 device_config = {
3          'miso':19,
4          'mosi':27,
5          'ss':18,
6          'sck':5,
7          'dio_0':26,
8          'reset':14,
9          'led':2,
10 }
```

#### ESP und LoRaWAN Konfiguration des ESP

In den *lora\_parameters* wird die Einstellung des LoRa Chips vorgenommen und verwaltet. Hier wird die Frequenz, der Sendekanal und der Aufbau der LoRa Botschaft definiert.

```
lora_parameters = {
      'frequency': 868E6, #LoRa Frequenz Europa
2
      'channel': 0, #Der verwendete Sendekanal
      'tx_power_level': 2, #Sendeleistung
5
      'signal_bandwidth': 'SF7BW125', #Bandbreite
      'spreading_factor': 7, #Spreizfaktor
      'coding_rate': 5,
8
      'preamble_length': 8,
      'implicit_header': False,
      'sync_word': 0x12,
10
      'enable_CRC': True, #muss auf True gesetzt werden!
11
      'invert IQ': False.
12
13 }
```

#### ESP und LoRaWAN Konfiguration des ESP

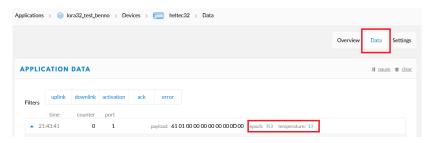
Damit die Daten des ESP auch in der erstellten Applikation ankommen, müssen die benötigten Adressen und Schlüssel aus der Application-Overview noch eingefügt werden.

(An die eigenen Parameter anpassen!)

## Werte per LoRaWAN versenden Beispielprogramm

Die Datei boot.py enthält ein kleines Beispielprogramm, das zufällige Temperaturwerte generiert und diese zusammen mit einem Zeitstempel verschickt.

Die Werte werden in der TTN-Console unter Data angezeigt:



→ Payload wird mithilfe des Decoders dekodiert!

#### Projekt: Alarmanlage Werte per Knopfdruck verschicken

#### Aufgabenstellung:

- Eingang der Alarmanlage ist ein digitales Signal (ESP Eingang Pin0 wird mit GND verbunden)
- Wenn die Alarmanlage auslöst soll eine LoRaWAN-Botschaft gesendet werden
- Jedes mal wenn die Alarmanlage auslöst soll ein Zähler hochgezählt werden

## Storage Integration Werte speichern

Normalerweise verschwinden die übermittelten Werte nach kurzer Zeit wieder aus der TTN-Console. Um die Werte zu sichern, verwenden wir die **Storage Integration**, mit deren Hilfe sich Werte bis zu 7 Tage speichern und auch abrufen lassen.

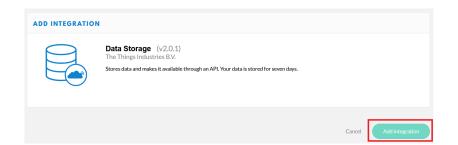
## Storage Integration Integration einbinden

Unter Integrations auf add intregration klicken:



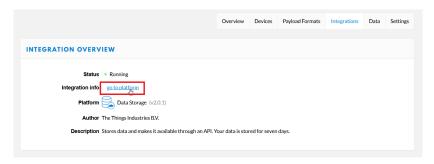
### Storage Integration Integration einbinden

Anschließend die *Data Storage Integration* auswählen und auf *Add integration* klicken:



### Storage Integration Daten abrufen

In der Integration-Overview auf go to platform klicken:



Auf *Authorize* klicken und den Access Key aus der Application Overview eingeben. Anschließend hat man verschiedene Möglichkeiten, die Daten abzurufen.

### Storage Integration Daten abrufen

Über einen Klick auf *Try it out* können Datensätze abgerufen werden, der ensprechende Curl-Befehl für die Kommandozeile kann kopiert werden (mit " statt ')

```
"Microsoft Windows [Version 10.0.19841.572]
(c) 2020 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Benno>c:\Users\Benno>c:\""https://lona32_test_benno.data.thethingsnetwork.org/api/v2/query/last=2d"
[("device_id":"heltec32", __epoch":353] "raw":"YQEAAAAAAAAAAA—=" __temperature":13, "time":"2020-10-20119:43:41.9815269282")]

C:\Users\Benno>
```

### Storage Integration Daten abrufen mit Python

```
Imports...
2
3 context = ssl.create_default_context()
  context.check hostname = False
5
6 url = 'https://lora32_test_benno.data.thethingsnetwork.
      org/api/v2/query'
7
  args = '?last=7d'
g
10 access_key = '<access-key>'
  headers = {
          'Accept': 'application/json',
12
           'Authorization': 'key ' + access_key
13
14
15 req = urllib.request.Request(url + args, headers=headers)
16
17 # Datenverarbeitung...
```