IoT in der Praxis mit LoRaWAN

Akademie Dillingen

20. Nationaler Akademietag - 23. April 2021

Prof. Dr. Hubert Högl (Hochschule Augsburg)

Maximilian Munz, Franz Refle, Wolfgang Trittner, Manfred Wolf (Bürgernetz Dillingen)

Zeitplan

Kleine Übersicht zu IoT und LoRaWAN (ca. 20 Minuten)

Vier Praxisteile P1 - P4 (je ca. 10 bis 15 Minuten)

P1. "CO2 Ampel" als Beispiel eines IoT Gerätes (Hardware)

P2. Programmierung der CO2 Ampel mit freier Software

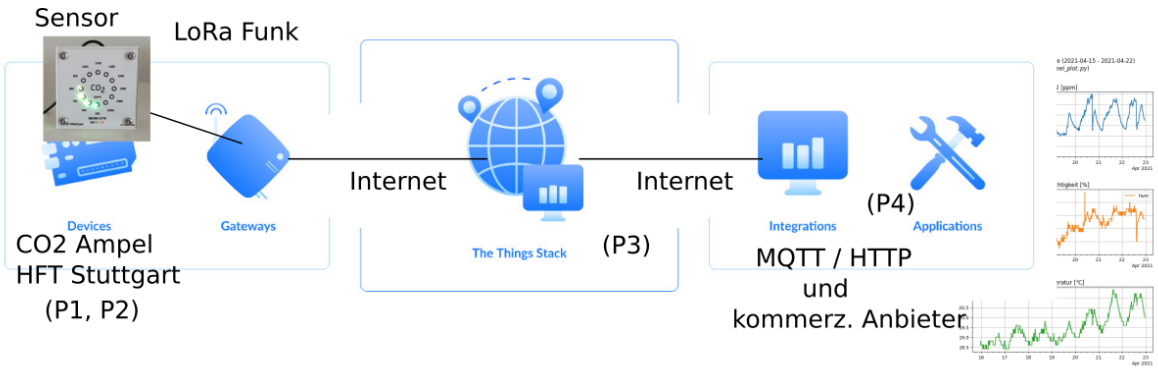
P3. CO2 Ampel im LoRaWAN Netzwerk bekannt geben

P4. Zugreifen auf die Sensordaten

Bei Fragen gerne zwischendurch beim Moderator melden.

Vom Sensor zur Auswertung

Quelle: https://www.thethingsindustries.com/docs/



Übersicht zu IoT und LoRaWAN

Welche **Dinge** und **Anwendungen**?

- Umweltsensoren (Temperatur, Luftfeuchte, CO2, Feinstaub, ...)
- Öffentliche Infrastruktur (Beleuchtung, Mülltonnen, Hundeklos, Nagetierfallen, ...)
- Verbrauchszähler (Gas, Wasser, Strom)
- Individuelle Aufgaben, z.B. Wasserstandsmessung im Brunnen, Bodenfeuchte auf Sportplatz

Übliche **IoT Merkmale**

- Sehr kleine Datenraten (einige 100 Bytes/Tag)
- Jahrelanger wartungsfreier Betrieb aus Batterie oder autonom mit *Energy Harvesting*
- Grosse Reichweite im Kilometer-Bereich

LPWAN

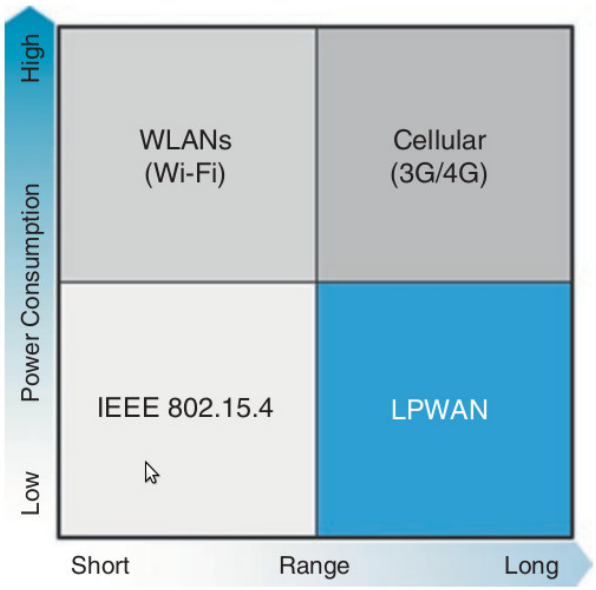
Low Power Wide Area Network [1]

Low Power bedeutet "Low Energy"!

Drei typische Repräsentanten:

- LTE Cat NB1, "NB-IoT"
- Sigfox "0G" [4], https://sigfox.de
- LoRaWAN

(Das Bild stammt aus Rayes/Salam 2019)



Vergleich einiger technischer Daten

NB-IoT	Sigfox	LoRa/LoRaWAN

Band	700-900 MHz	868 MHz	868 MHz
Bandbreite	200 kHz	200 kHz	125-500 kHz
Datenrate	128 kbit/s	100 bit/s	290 b/s - 50 kb/s
Reichweite	< 10 km	2-20 km	2-10 km
Datenpaket	> 1000 byte	12 byte	51 byte
Stromverbrauch TX	...	...	40 mA
Stromverbrauch RX	...	...	10 mA
Standby	mA	uA	uA
Infrastruktur	Netzbetreiber	Netzbetreiber Freiwillige	

Wie wenig Energie ist *Low Energy*?

Beispiel (aus [5]): AA Zelle: 1,2V, 2000 mAh, 2,4 Wh Energiegehalt (ca. 8640 Joule)

Geplante Laufzeit 10 Jahre. Reichen 3 Stück AA Zellen?

Ergebnis

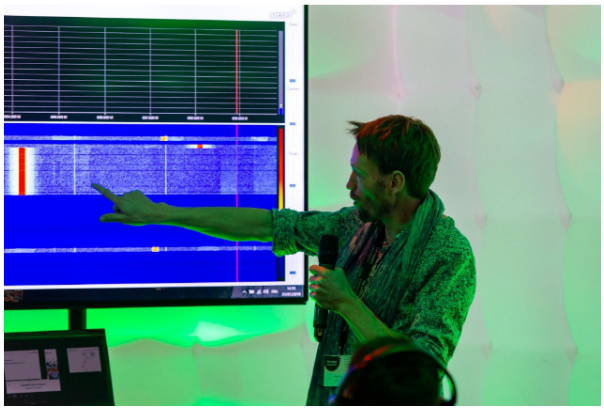
- Bei LoRa und Sigfox sind mit 3 x AA in Summe ca. 51.840 Übertragungen möglich
- 10 Jahre sind 3650 Tage, ca. 14 Übertragungen am Tag.
- Bei widrigen Umständen nur 1/4 der Zeit: 2,5 Jahre.

LoRa

- **Lo**ng **Ra**nge
- Nicolas Sornin (Bild) und Olivier Seller
- Gründung von **Cycleo**, 2009 (Frankreich)

  Aufgekauft durch **Semtech** (USA) 2012
  https://www.semtech.com/lora

  Liefern Bausteine **SX12xx**



- Modulation **Chirp Spread-Spectrum**, Patent 2014
- Signal kann weit unter dem **Rauschpegel** sein
- **Physikalische Schicht** im ISO/OSI Modell

- Kein "Listen-before-Talk", nur max **1% Duty Cycle**
- Geräte aufgeteilt in **Class A**, **Class B** und **Class C**
- **Uplink** und **Downlink**

Lit.: [6], [8]

LoRaWAN

- **Wide Area Network** mit LoRa
- **DataLink/Netzwerk-Schichten** im ISO/OSI Modell
- Slogans: **YOU ARE THE NETWORK** / **LET'S BUILD THIS THING TOGETHER**

  Basisdemokratisches Modell: Komplette Infrastruktur in der Hand der Bürger.
- Komplette Netzwerksoftware ist **Open-Source** (Go), darf beliebig genutzt werden zum Aufbau von eigenen Netzen.

  The Things Stack https://github.com/TheThingsNetwork/lorawan-stack
- LoRaWAN **Spezifikation V1.1** https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/
- **The Things Network** (TTN) https://www.thethingsnetwork.org als weltweites öffentliches LoRaWAN

Lit.: [6], [7]

LoRaWAN Infrastruktur

- End Nodes
- Gateways
- Network Server mit Konsole
- Application Server
- Schnittstellen zum Application Server
  - MQTT
  - HTTP

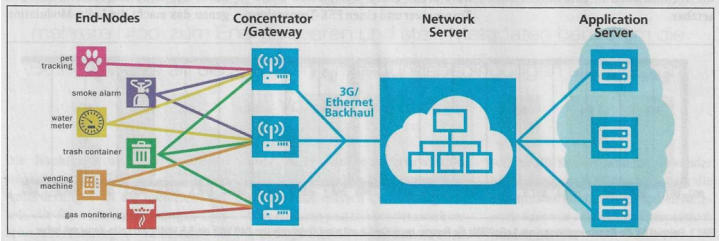


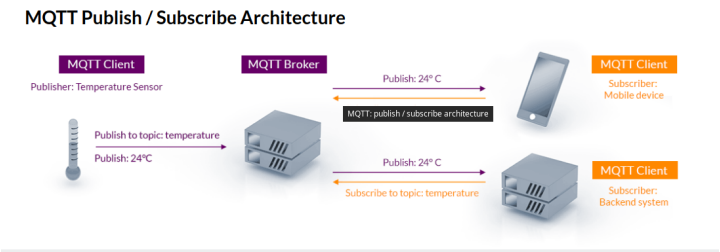
Bild 3. Zentraler Baustein eines LoRaWAN-Netzwerkes ist der Netzwerkserver. Er leitet die Daten von und zu den Funksensoren weiter und steuert auch das LoRaWAN, um das Funknetzwerk an veränderte Umgebungsbedingungen anzupassen. (Bild: Semtech)

(Bild aus M. Fink, Elektronik 7.2021)

MQTT

- Zentraler Server - "Broker"
- Effizient, sicher
- Publish/Subscribe

- Topics
- Payload
- Freie Software
  - Paho MQTT Client
  - Eclipse Mosquitto Broker
- https://mqtt.org



Integrations

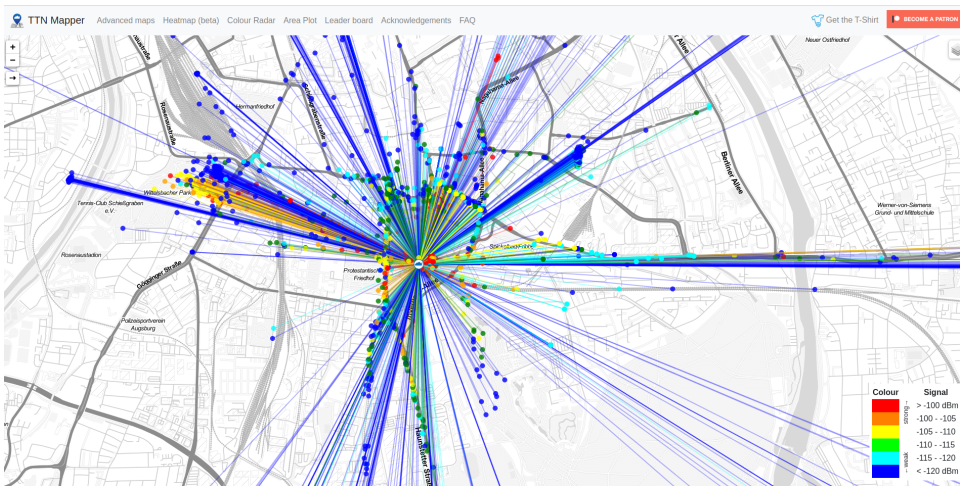
Datenschnittstellen

- HTTP Integration
- MQTT
- Data Storage

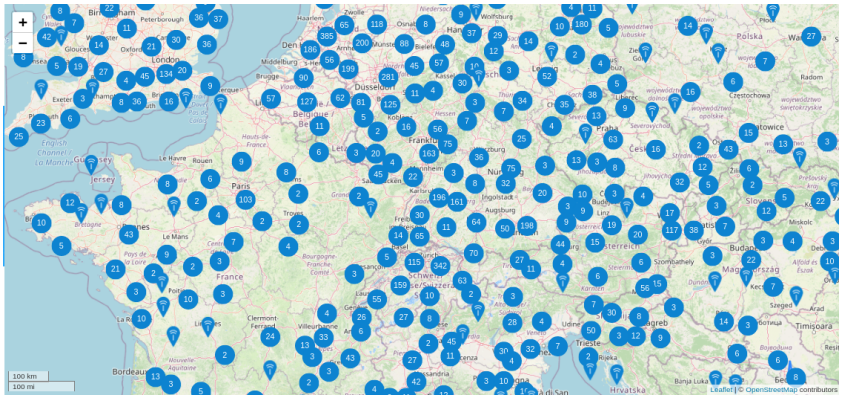
Geschäftsmodelle von Dritten

- https://www.allthingstalk.com
- https://thingspeak.com
- https://mydevices.com
- https://www.loracloud.com (Collos / Geolocation)
- https://tago.io

Beispiel: Integration TTN Mapper



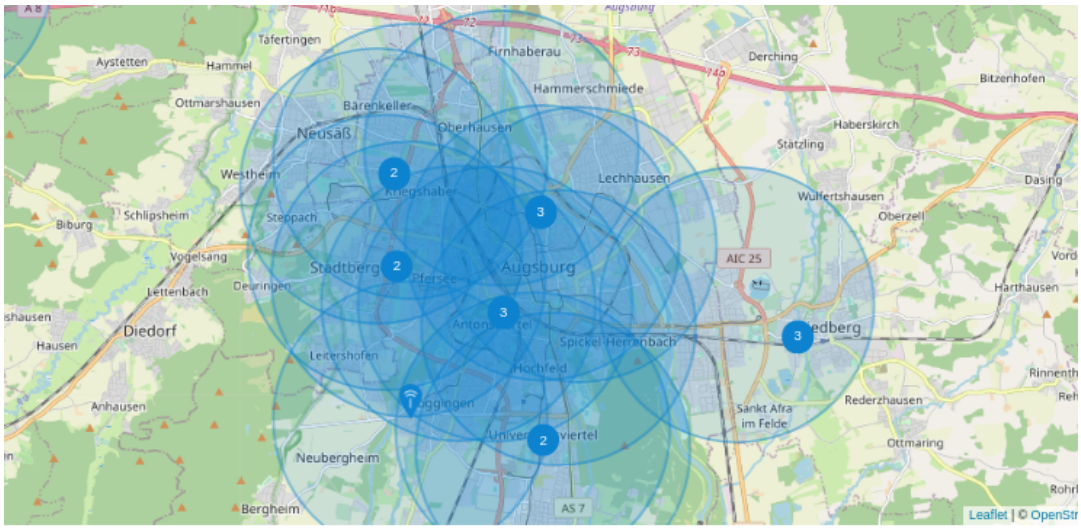
TTN Europa



TTN Community Augsburg

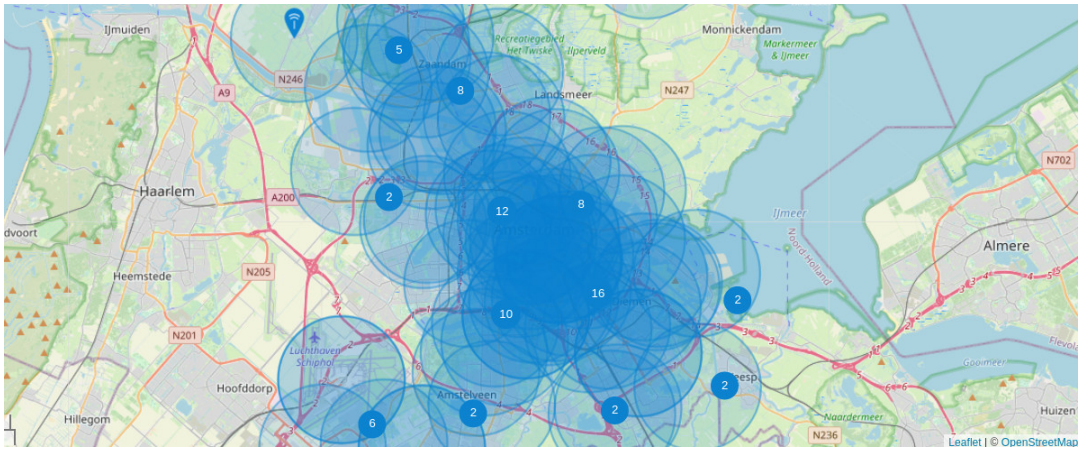
http://hhoegl.informatik.hs-augsburg.de/hhwiki/ttn

TTN Community Augsburg (2)



Sehr grosse TTN Communities

- Ulm, Stuttgart, Berlin, Zürich, Amsterdam (Bild), viele weitere ...



Praxisteil 1 - Hardware der CO2 Ampel

(Wolfgang Trittner)

- https://hpsrv-cloud.informatik.hs-augsburg.de/user/guest/notebooks/01wolfi.ipynb

Praxisteil 2 - Software für CO2 Ampel kompilieren

(Hubert Högl)

- Live Vorführung am Rechner mit der Arduino Umgebung

Praxisteil 3 - Gerät auf LoRaWAN Server (TTN) eintragen

(Franz Refle)

- Aktuell Übergang von TTN Server V2 auf V3
- V2 Konsole: https://console.thethingsnetwork.org
- V3 Konsole: https://eu1.cloud.thethings.network/console/

Praxisteil 4 - Auf die Daten zugreifen

(Hubert Högl)

- Data Storage Integration mit Plot
- MQTT
- HTTP

Data Storage Integration mit Plot

https://www.thethingsnetwork.org/docs/applications/storage/index.html

- `co2ampel_plot.py` https://github.com/huberthoegl/ttn-democode

- Daten einer Woche über OpenAPI holen (RESTful API, https://swagger.io/specification)

- In Pandas DataFrame umwandeln
- Mit Matplotlib grossen Plot mit drei Subplots machen (CO2, Feuchte, Temperatur)

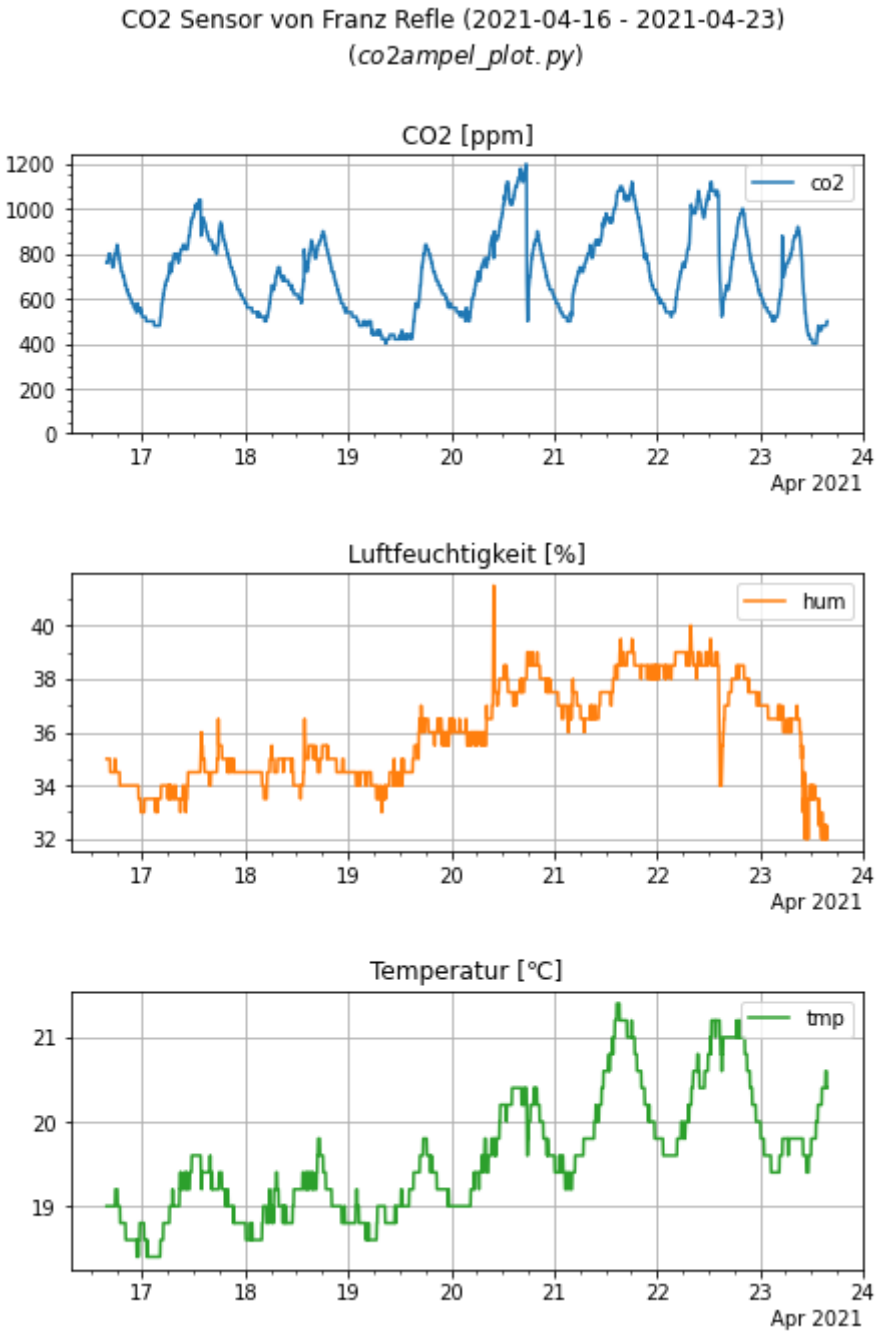
In [31]:

```
# co2ampel_plot.py

"""
This script gets data from the "fr_co2ampel_hft" LoRaWAN application by the
"Data Storage Integration".  The data is stored at the TTN server for 7 days.

Further information:

- https://www.thethingsnetwork.org/docs/applications/storage/api/index.html
- https://console.thethingsnetwork.org/applications/fr_co2ampel_hft/integrati
- https://fr_co2ampel_hft.data.thethingsnetwork.org
- https://fr_co2ampel_hft.data.thethingsnetwork.org/swagger.yaml

Authors:
Franz Refle, 2021 (original work)
Hubert Högl, 2021, <Hubert.Hoegl@hs-augsburg.de>
"""

import os
import requests
import sys
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator
from matplotlib.dates import DayLocator, HourLocator, DateFormatter, drange
import matplotlib.dates as mdates

url = "https://fr_co2ampel_hft.data.thethingsnetwork.org/api/v2/query/co2ampe
args = '?last=7d'

# You need an access key to get the data. Go to
# https://console.thethingsnetwork.org/applications/fr_co2ampel_hft
# at the bottom you see "ACCESS KEYS". Set environment variable ACCESSKEY to
# this key with the command "export ACCESSKEY=..."
if "ACCESSKEY" not in os.environ:
    raise(ValueError('environment variable ACCESSKEY not set'))
else:
    access_key = os.environ["ACCESSKEY"]

# print(access_key)

headers = {'Accept': 'application/json', 'Authorization': 'key ' + access_key

# Get data by Swagger UI (OpenAPI)
try:
    response = requests.get(url + args, headers=headers)
except OSError as e:
    print("Error: {0}".format(e))
    sys.exit(0)

if response.status_code == 200:
    print("Status 200, OK")
    data = response.json()
else:
    print("Error (response.status_code is {})".format(response.status_code))
    sys.exit(0)
```

```
# Time format in data: 2021-03-30T13:53:03.742880288Z
# the 9 digit nsec fraction can not be parsed with strptime(). Cut off the la
# four chars to get a microsecond fraction.
begin_date = data[0]['time'][:-4]
end_date = data[-1]['time'][:-4]

fmt = "%Y-%m-%dT%H:%M:%S.%f" # %f is a 6 digit microsecond fraction
dt1 = datetime.strptime(begin_date, fmt)
dt2 = datetime.strptime(end_date, fmt)
d1 = dt1.date()
d2 = dt2.date()

df = pd.DataFrame(data)

# df DataFrame:
#    co2   device_id           hum   raw           time                     tmp
#0  1040  co2ampelbndlg_dev01  38.0  NJdM  2021-03-30T10:22:01.460562325Z 20.
#1  1020  co2ampelbndlg_dev01  37.5  M5hL  2021-03-30T10:27:10.143553097Z 20.
#2  1020  co2ampelbndlg_dev01  37.5  M5hL  2021-03-30T10:32:18.711297525Z 20.
#...

# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_dateti
df['time'] = pd.to_datetime(df['time'])

ts = df.set_index('time')

cm = 1/2.54  # inch to cm

# Place 3 subplots in one large plot on a A4 page
fig, axes = plt.subplots(3, figsize=(18*cm, 26*cm))
# print("fig:", type(fig), fig)

# Put format placeholders in single {...}, LaTeX commands in double {{...}}!
fig.suptitle(r"""CO2 Sensor von Franz Refle ({begin} - {end})
   ($\it{{co2ampel\_plot.py}}$)
""".format(begin=d1, end=d2))

# ~~~ snippet from https://matplotlib.org/stable/gallery/ticks_and_spines/dat
for nn, ax in enumerate(axes):
    locator = mdates.AutoDateLocator()
    formatter = mdates.ConciseDateFormatter(locator)
    formatter.formats = ['%y',  # ticks are mostly years
                         '%b',       # ticks are mostly months
                         '%d',       # ticks are mostly days
                         '%H:%M',    # hrs
                         '%H:%M',    # min
                         '%S.%f', ]  # secs
    # these are mostly just the level above...
    formatter.zero_formats = [''] + formatter.formats[:-1]
    # ...except for ticks that are mostly hours, then it is nice to have
    # month-day:
    formatter.zero_formats[3] = '%d-%b'

    formatter.offset_formats = ['',
                                '%Y',
                                '%b %Y',
                                '%d %b %Y',
                                '%d %b %Y',
                                '%d %b %Y %H:%M', ]
    ax.xaxis.set_major_locator(locator)
    ax.xaxis.set_major_formatter(formatter)

    # need minor ticks at multiples of 6 hours
    # https://www.geeksforgeeks.org/matplotlib-axis-axis-set_minor_locator-fu
```

```
    ax.xaxis.set_minor_locator(HourLocator(range(0, 25, 6)))
# ~~~

axes[0].set_title("CO2 [ppm]")
axes[1].set_title("Luftfeuchtigkeit [%]")
axes[2].set_title("Temperatur [°C]")

ts.plot(subplots=True, ax=[axes[0], axes[1], axes[2]], grid=True)

Ticks = axes[0].get_yticks()
axes[0].set_yticks(np.arange(0, 1400, 200))
Ticks = axes[0].get_yticks()
# print("Ticks[0] =", Ticks)
axes[0].yaxis.set_minor_locator(MultipleLocator(50))
axes[0].set_xlabel("") # remove "time"
# rotation not needed for concise date format
# for label in axes[0].get_xticklabels():
#    label.set_rotation(20)
#    label.set_horizontalalignment('right')

Ticks = axes[1].get_yticks()
# print("Ticks[1] =", Ticks)
axes[1].yaxis.set_minor_locator(MultipleLocator(1.0))
axes[1].set_xlabel("") # remove "time"

Ticks = axes[2].get_yticks()
# print("Ticks[2] =", Ticks)
axes[2].yaxis.set_minor_locator(MultipleLocator(1.0))
axes[2].set_xlabel("") # remove "time"

# https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.subplots_adjust.
# wspace = width reserved, hspace = height reserved
plt.subplots_adjust(wspace=0.5, hspace=0.5)

plotfile = "plot.jpg"
#plt.savefig(plotfile)
#print("see", plotfile)
plt.show()
```

```
Status 200, OK
```

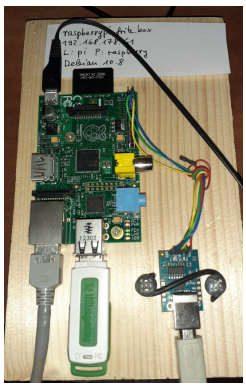


MQTT / InfluxDB

- co2ampel_client.py https://github.com/huberthoegl/ttn-democode
- InfluxDB https://www.influxdata.com

  Open Source! https://github.com/influxdata/influxdb

  Ideal für die Abspeicherung und Suche von Sensordaten.



Open-Source Werkzeuge für den Workshop

- Linux!

- Jupyter Notebook https://jupyter.org
- RISE - Folienpräsentation im Browser https://rise.readthedocs.io/en/stable/
- Matplotlib https://matplotlib.org
- Numpy https://numpy.org

Jupyter, *Matplotlib* und *Numpy* stammen aus dem riesigen Open-Source Angebot der Programmiersprache *Python* (https://www.python.org). Wer sich damit befassen möchte, nimmt am besten die freie *Anaconda* Distribution, siehe https://www.anaconda.com/products/individual.

Literatur/Links

[1] https://de.wikipedia.org/wiki/Low_Power_Wide_Area_Network
[2] https://en.wikipedia.org/wiki/LTE-M
[3] https://www.iis.fraunhofer.de/de/ff/lv/net/telemetrie.html
[4] https://de.wikipedia.org/wiki/Sigfox
[5] https://www.elektronikpraxis.vogel.de/lpwan-technologien-im-vergleich-a-832893
[6] https://www.elektronikpraxis.vogel.de/lorawan-im-detail-so-arbeitet-die-iot-funktechnik-a-836031
[7] https://lora-alliance.org/about-lorawan
[8] http://hhoegl.informatik.hs-augsburg.de/doc/lora-lit18.pdf
[9] https://lora-alliance.org/about-lorawan