Erste Schritte mit dem

DNP/AISS1 Machine Learning (ML) Starterkit

Preview Version 0.4 (05.2019)

Der Lieferumfang des DNP/AISS1-Starterkits

Zum Lieferumfang des DNP/AISS1-Starterkits gehören die folgenden Komponenten. Beachten Sie vor dem Einsatz des Starterkits unbedingt die beigefügte Dokumentation.

1x vorkonfiguriertes DNP/AISS1 mit voreingestellten IP-Adressen für die LAN-Schnittstelle

1x Ethernet-LAN-Kabel

1x 12 VDC oder 24 VDC-Steckernetzteil für die DNP/AISS1-Baugruppe

1x Verbindungskabel 30 cm

1x Taschenbuch „Machine Learning – kurz und gut“ von O’Reilly

1x gedruckte Bedienungsanleitung für die ersten Schritte der Inbetriebnahme

Welche Voraussetzungen sind erforderlich?

Für die Nutzung des DNP/AISS1-Starterkits benötigen Sie einen PC mit Internetzugang und eine lokale Ethernet-LAN-Verbindung zum DNP/AISS1. Des Weiteren muss der PC mindestens die hier folgenden Voraussetzungen erfüllen:

- Es muss ein HTML5-standardkonformer Webbrowser (z. B. Chrome oder Firefox) auf dem PC zur Verfügung stehen.

- Es muss ein Editor zum Bearbeiten von Quellcodes auf dem PC installiert werden. Siehe zum Beispiel https://notepad-plus-plus.org/.

- Es muss ein FTP-Client auf dem PC installiert werden, um Dateien vom PC aus in das Dateisystem der DNP/AISS1-Baugruppe zu übertragen. Siehe z. B. https://filezilla-project.org/.

- Die LAN-Verbindung zwischen PC und DNP/AISS1 sollte möglichst durch **keine** aktive Firewall geschützt sein. Falls doch eine Firewall existiert, müssen neben dem TCP-Port 80 (HTTP) auch der TCP-Port 1880 (Node-RED-UI) und TCP-Port 4200 (Shell-In-a-Box Service, Web Console) freigeschaltet werden.

Auf dem DNP/AISS1-Starterkit ist Node-RED vorinstalliert und kann von Ihrem PC aus über die dafür vorgesehene Webschnittstelle per Browser genutzt werden. Daher folgende Empfehlung:

- Machen Sie sich mit der Benutzung von Node-RED vertraut. Das Internet bietet hierfür unzählige Möglichkeiten. Siehe hierzu auch https://nodered.org/docs/getting-started/first-flow.

- Machen Sie sich des Weiteren mit den Grundgedanken des Node-RED-Dashboards vertraut. Siehe hierzu https://flows.nodered.org/node/node-red-dashboard.

Falls Sie weitere Eigenschaften und Möglichkeiten des DNP/AISS1-Starterkits nutzen wollen, sind hinsichtlich der Firewall-Einstellungen ggf. weitere Ports freizuschalten und zusätzliche Softwareinstallationen auf dem PC erforderlich. Beachten Sie dazu die jeweiligen Hinweise in den entsprechenden Dokumenten.

Die erste Übung: Node-RED-Zugriff per Webbrowser

Bitte zuerst die Ethernet-Schnittstelle des PCs mit dem Ethernet-LAN-Port des DNP/AISS1 verbinden. Benutzen Sie hierfür bitte das LAN-Kabel aus dem Lieferumfang des Starterkits.

Der DNP/AISS1-LAN-Port ist ab Werk fest auf die statische IP-Adresse **192.168.0.126** voreingestellt. Für den LAN-Port Ihres PCs sollten Sie daher zum Beispiel die statische die IP-Adresse **192.168.0.1** verwenden.

Verbinden Sie nun die beiden Kabelenden des 12 VDC bzw. 24 VDC-Steckernetzteil aus dem Starterkit-Lieferumfang mit der 2-poligen-Versorgungsspannungsklemme des DNP/AISS1 (**Achtung:** Die rot gekennzeichnete Ader ist +12 bzw. +24 VDC). Stecken Sie danach das Steckernetzteil in eine Steckdose und warten Sie den Bootvorgang des DNP/AISS1 ab.

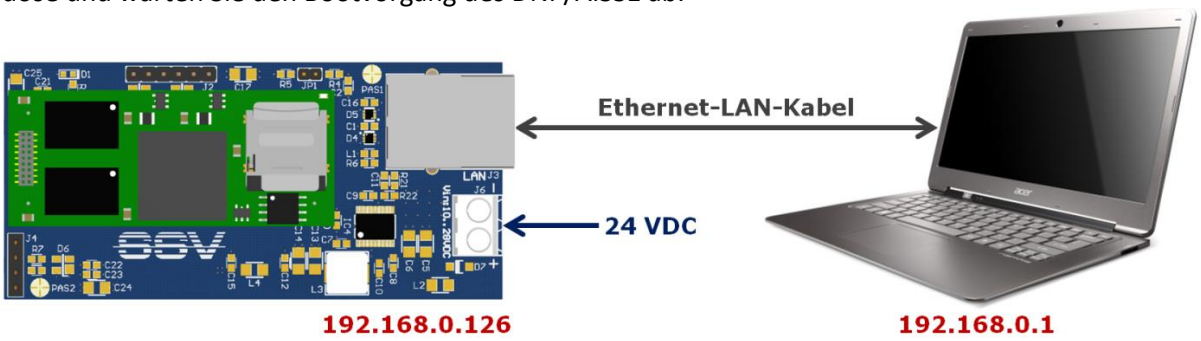
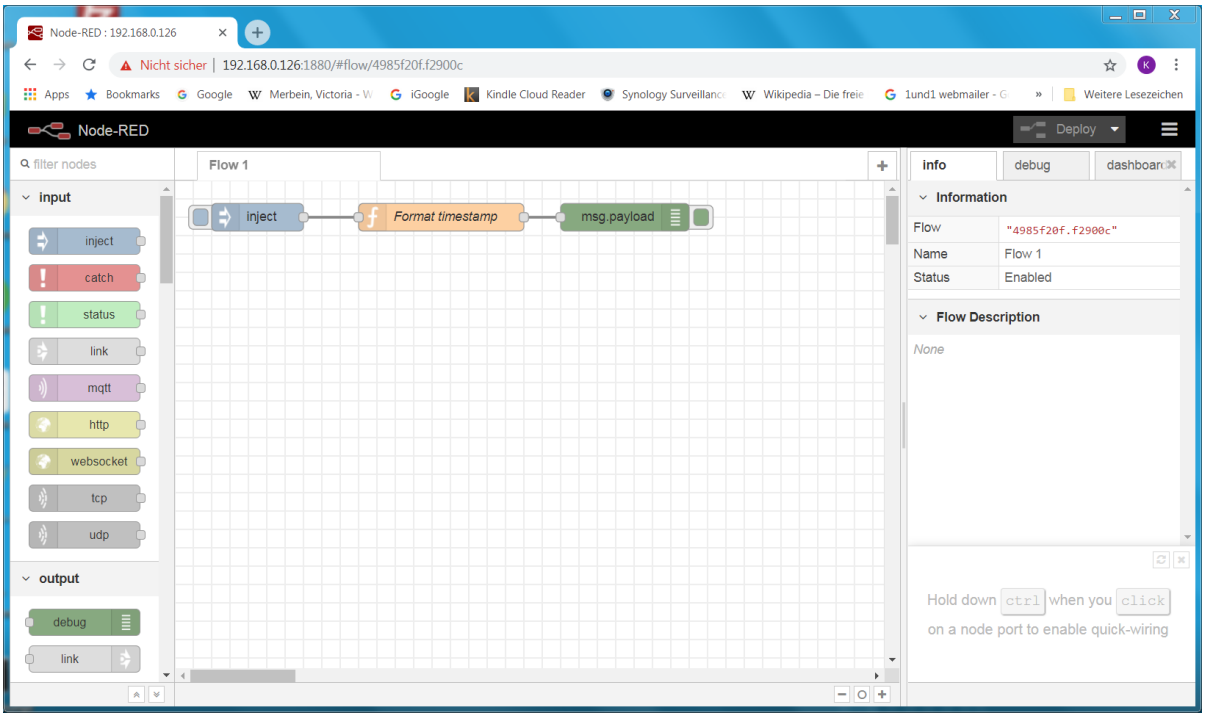


Abbildung: Der Ethernet-LAN-Port des DNP/AISS1 besitzt ab Werk die IP-Adressen 192.168.0.126. Ihr PC sollte die statische IP-Adresse 192.168.0.1 benutzen. Verkabeln Sie die Systeme wie in der Abbildung dargestellt. Verwenden Sie bitte das LAN-Kabel und das 12 bzw. 24 VDC-Steckernetzteil aus dem Lieferumfang des Starterkits.

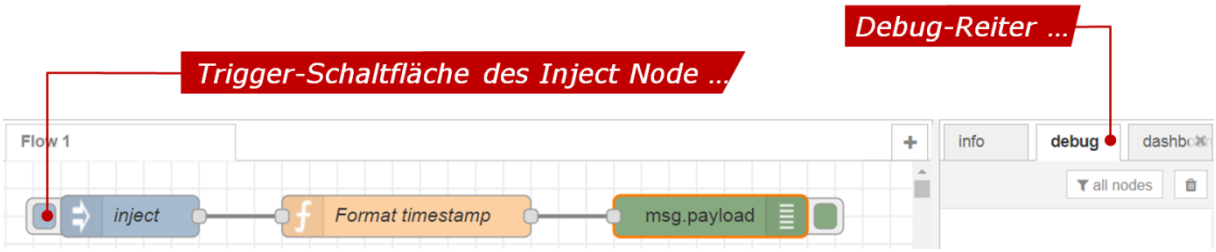
Rufen Sie auf Ihrem PC den Webbrowser auf. Geben Sie den Link http://192.168.0.126:1880 als URL ein. Dadurch wird Ihnen die Node-RED-Benutzeroberfläche (Node-RED-UI, siehe folgende Abbildung) angezeigt.



In der Arbeitsfläche des Node-RED-Fensters ist das erste Beispiel „Creating your first flow“ (siehe https://nodered.org/docs/getting-started/first-flow) bereits vorinstalliert. Dieses Beispiel besteht aus drei Nodes:

Ganz links ist ein *Inject Node*, in der Mitte ein *Function Node* und rechts ein *Debug Node* zu sehen. Mit einem doppelten Mausklick auf den jeweiligen Node können Sie sich die Konfigurationseinstellungen des ausgewählten Nodes anschauen.

Um die Funktion dieses Flows zu untersuchen, klicken Sie bitte zunächst auf den *Debug-Reiter* im rechten Teil des Node-RED-Fensters. Dadurch öffnet sich eine Ausgabefläche für die Meldungen der jeweiligen Debug Nodes eines Flows.



Klicken Sie nun in *Trigger-Schaltfläche* des Inject Nodes und beobachten sie dabei die jeweiligen Ausgaben des Debug Nodes im dazugehörenden Ausgabebereich.

Node-RED funktioniert zum einen nach dem Datenfluss-Prinzip und zum anderen ereignisgesteuert. Am Ausgang eines Nodes stehen als Folge eines Ereignisses (z. B. das Betätigen der Trigger-Schaltfläche eines Inject Nodes) jeweils Daten zur Verfügung. Die Ausgabe eines Nodes kann als Eingabe eines weiteren Nodes verwendet werden. Links in der Arbeitsfläche sind jeweils *Input Nodes* und rechts die *Output Nodes* eines Flows angeordnet. Zwischen Input Nodes und Output Nodes sind häufig Function Nodes platziert.

Die zweite Übung: Erster Zugriff auf Sensordaten

Nachdem der Webbrowser-Zugriff auf die Node-RED-Benutzeroberfläche des DNP/AISS1 als erster Schritt erfolgreich durchgeführt wurde, wollen wir per Node-RED auf ein Sensorelement zugreifen.

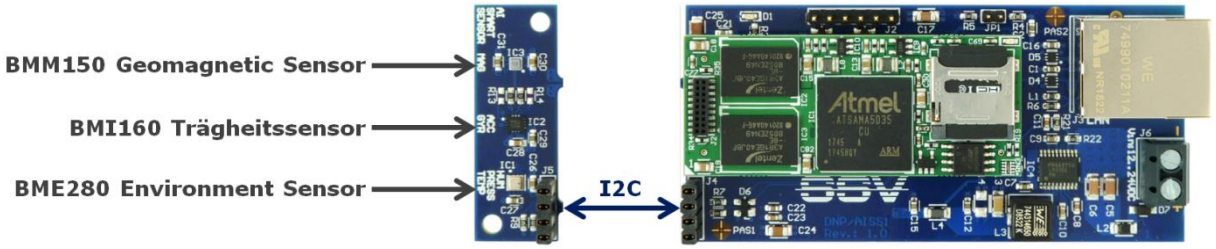


Abbildung: Der bei Bedarf abtrennbare Sensorkopf des DNP/AISS1 besitzt drei Bosch-Sensoren mit verschiedenen Sensorelementen: 1x BMM150 Geomagnetic Sensor, 1x BMI160 6-Achsen Inertial Measurement Unit (Trägheitssensor mit jeweils zwei triaxialen Sensorelementen) sowie 1x BME280 Environment Sensor mit drei Sensorelementen für Temperatur, Luftdruck und rel. Luftfeuchte.

Um ein Gefühl für die Zusammenhänge – vom Sensor bis zur Datenausgabe – zu entwickeln, werden wir nun einen Flow aus vier Nodes in einzelnen Schritten entwickeln.

Ziehen Sie zunächst die vier benötigten Nodes gemäß der folgenden Tabelle aus der Node-Palette am linken Rand in den Arbeitsbereich. Verdrahten Sie dann die Nodes untereinander. Konfigurieren Sie danach die ersten drei Nodes wie im folgenden Text beschrieben (**Achtung:** Um in die Konfigurations-einstellung eines Nodes zu gelangen, müssen Sie den entsprechenden Node „doppelklicken“).

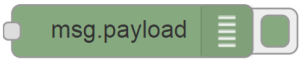
Node	Beschreibung
	*Inject Node*. Vergeben Sie in der Konfiguration den Namen „temp“. Wählen Sie als Payload „JSON“ und tragen Sie String „READ bme280 INPUT temp input\n“ in das jeweilige Eingabefeld ein. Beachten Sie dabei die Groß- und Kleinschreibung. Konfigurieren Sie des Weiteren ein Intervall von 5 Sekunden. Dadurch sendet der Inject Node alle 5 Sekunden einen JSON-Payload an den nachfolgenden Node.
	*TCP Request Node*. Vergeben Sie in der Konfiguration den Namen „iiod“ für diesen Node. Tragen Sie als Server-Adresse **127.0.0.1** und als Port **30431** in die dafür vorgesehenen Eingabefelder ein. Danach konfigurieren Sie bitte den Return-Modus des TCP Request Node auf „never – keep connection open“.
	*Function Node*. Vergeben Sie für diesen Node in der Konfiguration den Namen „format temp“. Übertragen Sie Code aus dem hier folgenden Listing 1 in das dafür vorgesehene Eingabefenster.
	*Debug Node*. Dieser Node benötigt in diesem Flow-Beispiel keinerlei spezielle Konfiguration.

Tabelle 1: Der Flow zu dieser Übung besteht aus insgesamt vier Nodes, von denen drei hinsichtlich der Anforderungen entsprechend konfiguriert werden. Der Function Node mit dem Namen „format temp“ benötigt darüber hinaus einige Zeilen JavaScript-Code, der im Listing 1 zu finden ist.

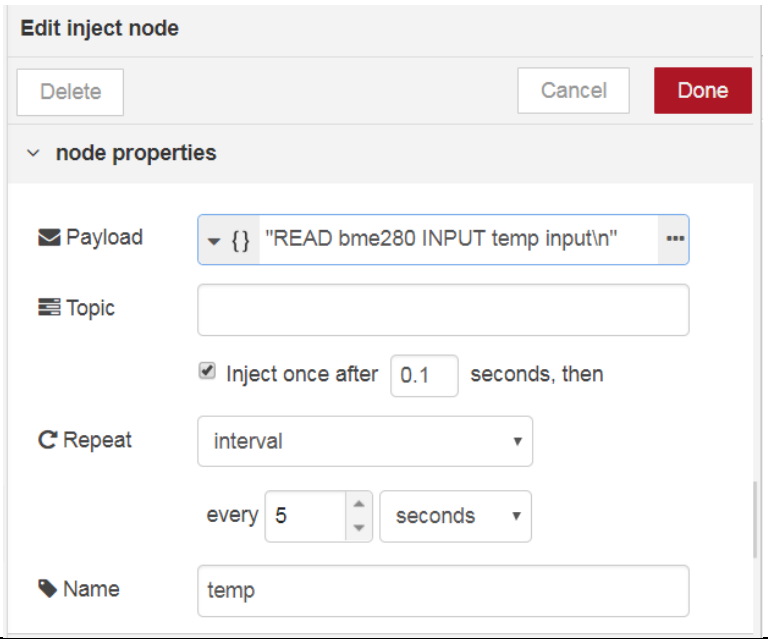
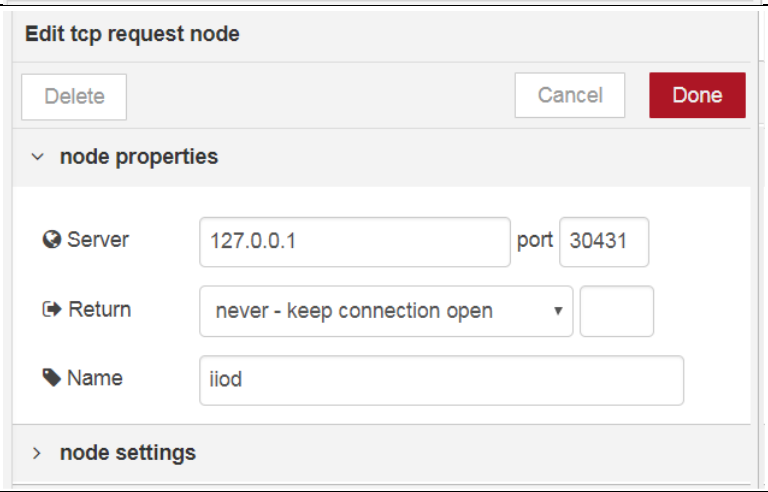
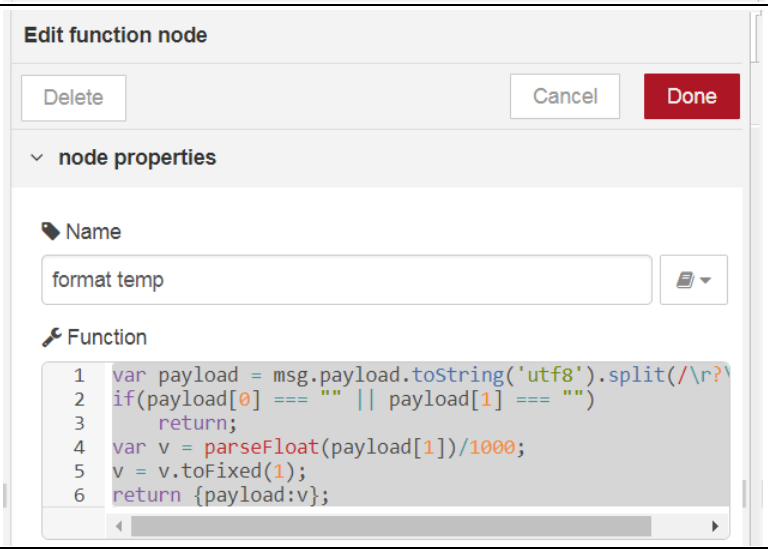
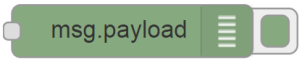
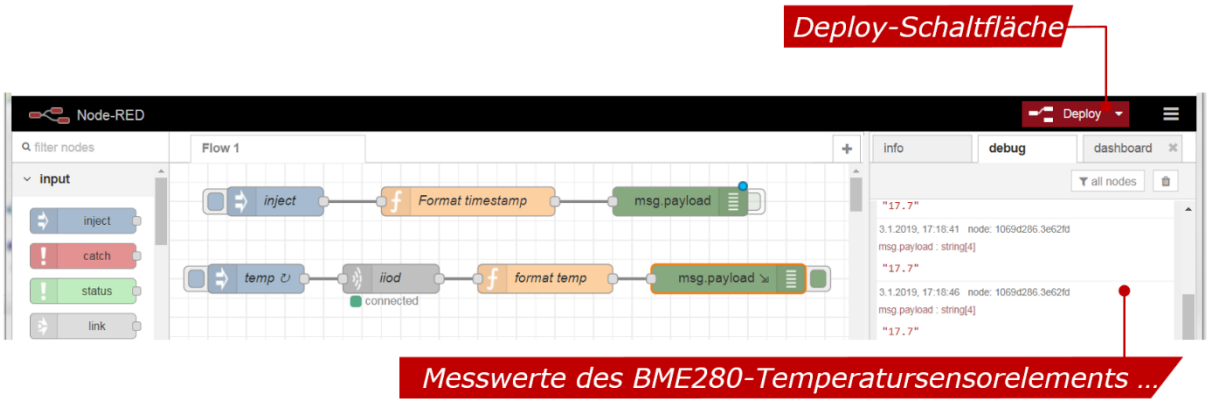
Node	Konfiguration
	
	
	
	*Debug Node*. Dieser Node benötigt in diesem Flow-Beispiel keinerlei spezielle Konfiguration.

Tabelle 2: Details der Konfigurationseinstellungen zu den drei Node-RED-Nodes für dieses Beispiel.

```
var payload = msg.payload.toString('utf8').split(/\r?\n/);
if(payload[0] === "" || payload[1] === "")
    return;
var v = parseFloat(payload[1])/1000;
v = v.toFixed(1);
return {payload:v};
```

Listing 1: Quellcode für den Function Node mit dem Namen „format temp“ (siehe Tabelle 1). Dieser JavaScript-Code sorgt dafür, dass die Rohdaten des BME280-Temperatursensorelements in eine geeignete Floating-Point-Zahl mit einer Nachkommastelle umgewandelt werden.

Wenn Sie die einzelnen Node-RED-Nodes dieses Beispiels gemäß den Anmerkungen aus den Tabellen 1 und 2 konfiguriert haben, können Sie den neu erstellten Flow zur Ausführung bringen. Betätigen Sie dazu bitte die *Deploy-Schaltfläche* im rechten oberen Bereich der Node-RED-Benutzeroberfläche:



In der zum Debug-Reiter gehörenden Ausgabefläche müssten Sie nun alle 5 Sekunden den aktuellen Messwert des BME280-Temperatursensorelements sehen.

Die dritte Übung: Hinzufügen eines Dashboards

Als nächstes wollen wir nun die Temperaturanzeige aus der zweiten Übung um eine graphische Oberfläche (Dashboard) mit zwei Anzeigeelementen erweiterten:

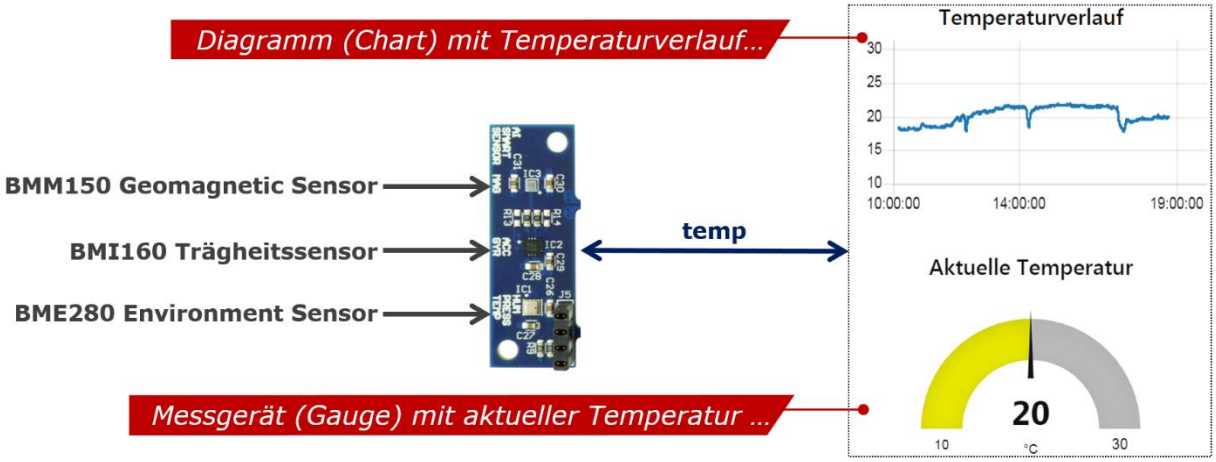
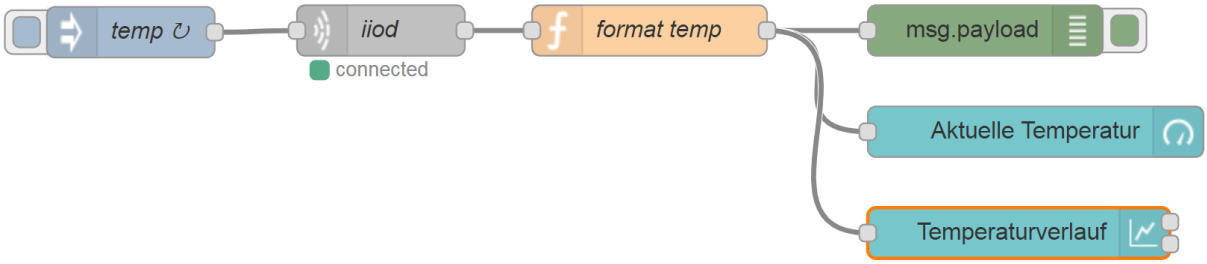
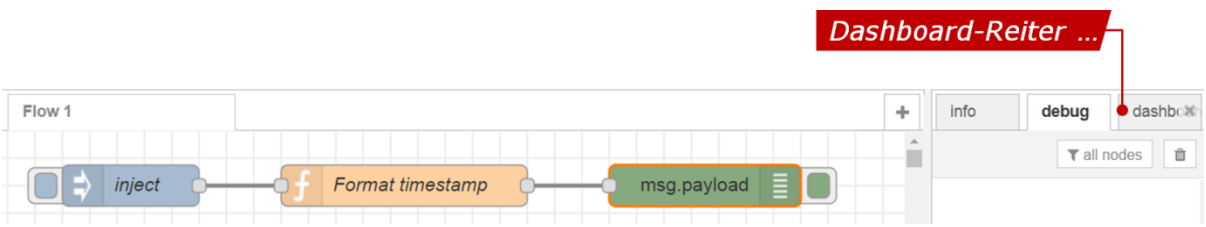


Abbildung: Durch die zweite Übung erhalten wir ca. alle 5 Sekunden den jeweils aktuellen Messwert des BME280-Temperatursensorelements. Mit Hilfe von Node-RED und der optionalen Dashboard-Erweiterung lassen sich die Temperaturmesswerte auch in einer graphischen Oberfläche darstellen.

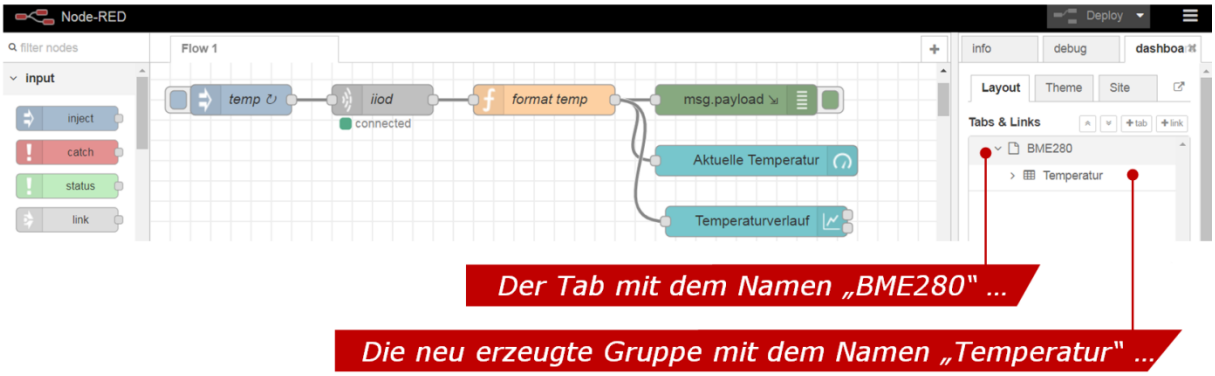
Um das Ziel dieser Übung zu erreichen, benötigen wir zwei zusätzliche Nodes aus der Node-RED-Dashboard-Erweiterung: je einen *Chart Node* und *Gauge Node*. Beide finden Sie in der *Dashboard-Gruppe* der Node-Palette am linken Rand des Arbeitsbereichs. Ziehen Sie beide Nodes in den Arbeitsbereich und verdrahten Sie Ihre Flow-Erweiterung gemäß der hier folgenden Abbildung (**Achtung:** Die Dashboard-Gruppe mit zusätzlichen Nodes existiert nur dann, wenn auch die Dashboard-Erweiterung auf dem DNP/AISS1 installiert wurde).



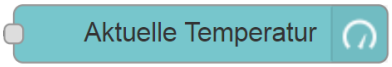
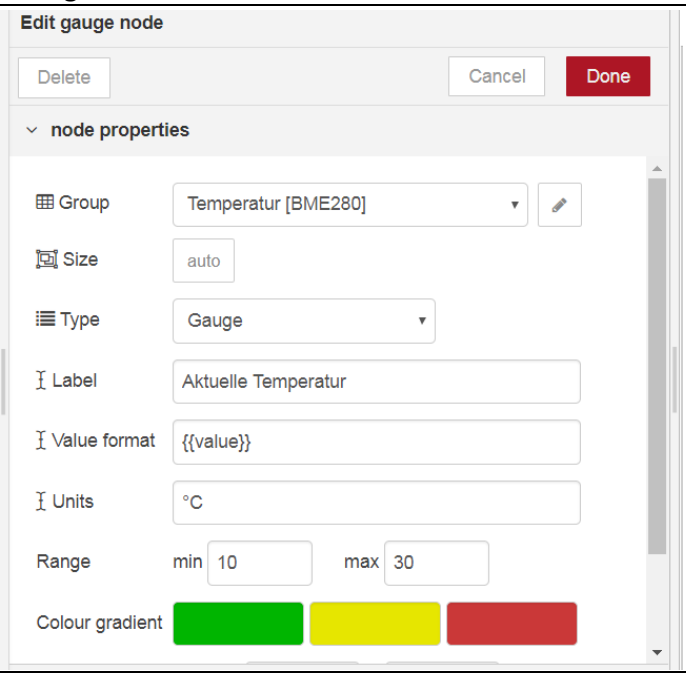
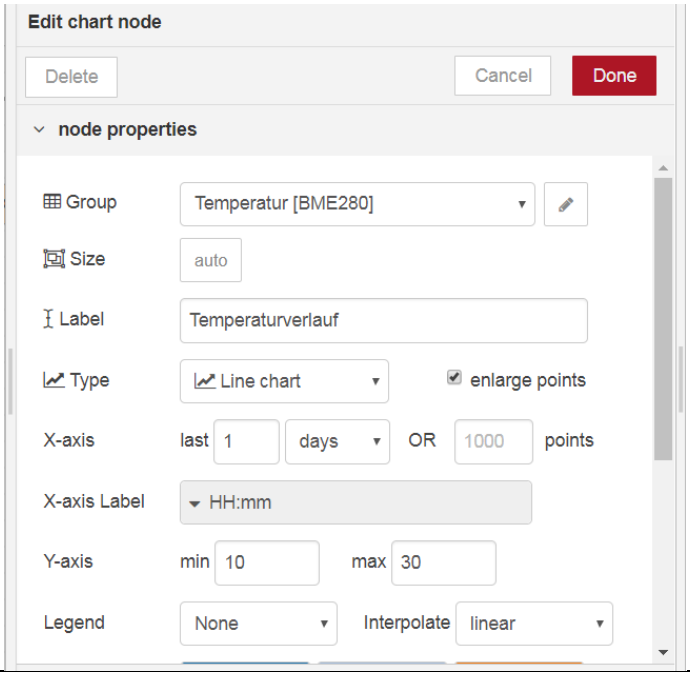
Konfigurieren Sie nun zunächst die grundlegenden Dashboard-Eigenschaften. Betätigen Sie hierzu als erstes den *Dashboard-Reiter* im rechten Teil des Node-RED-Fensters.



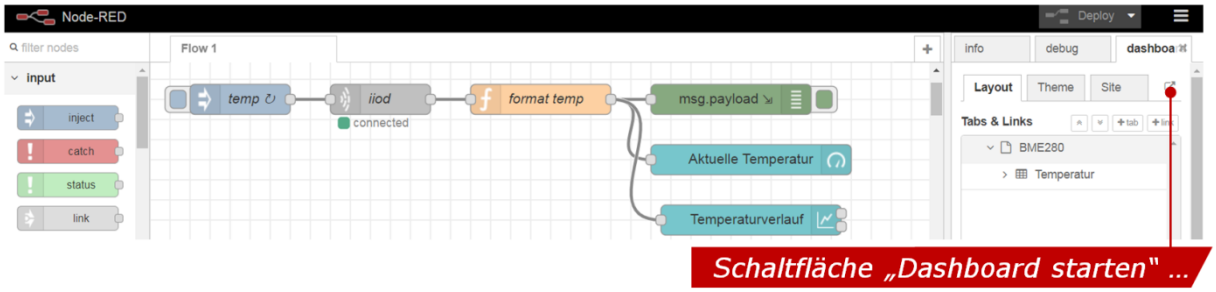
Benennen Sie dann den standardmäßig vorhandenen Tab in *BME280* um und erzeugen Sie unter diesem Tab eine neue Gruppe mit dem Namen *Temperatur* (Beachten Sie bitte die jeweilige Edit-Funktion). Durch die anschließend folgende Node-Konfiguration (**Achtung:** Beide Nodes jeweils „doppelklicken“ …) werden die Dashboard-Nodes der neuen Gruppe zugeordnet.



Konfigurieren Sie nun die beiden neuen Dashboard-Nodes mit den Vorgaben der folgenden Tabelle:

Node	Konfiguration
	
	

Zum Start des Dashboards betätigen Sie nun per Mausklick die entsprechende Schaltfläche (siehe die folgende Abbildung). Danach können Sie das neu entwickelte Dashboard per Webbrowser betrachten.



Achtung: Bitte beachten Sie unbedingt, dass der Temperaturverlauf im Chart Node des Dashboards nur bei einer laufenden Read Time Clock richtig dargestellt wird. Siehe hierzu auch *Anhang 2: Uhrzeit manuell setzen.* Stellen Sie diese Uhr für diese Übung manuell mit den jeweils aktuellen Datum- und Zeitangaben.

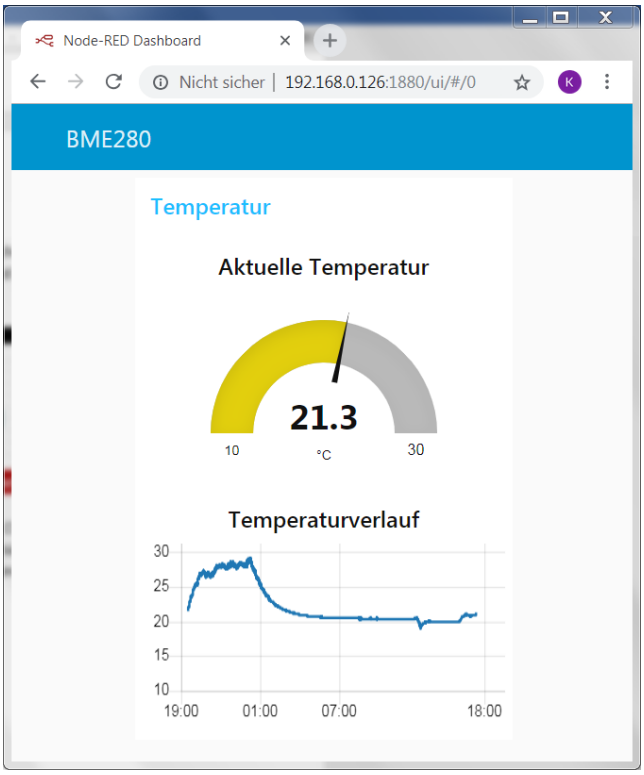


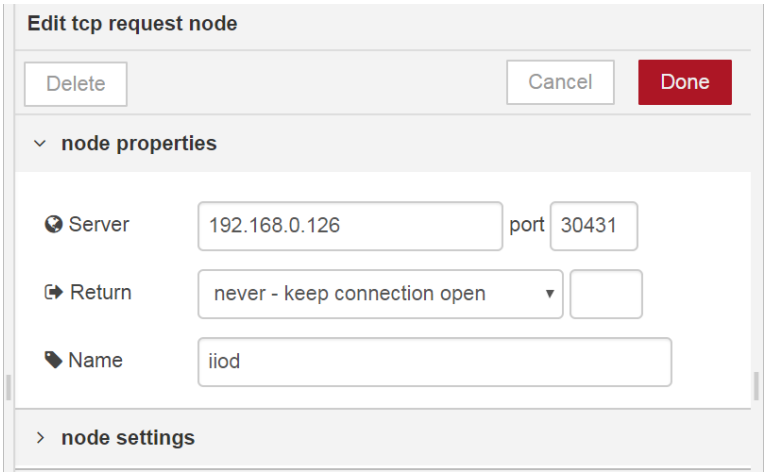
Abbildung: Ein Node-RED-Dashboard wird in einem eigenen Browser-Fenster angezeigt. Das Fenster kann durch Betätigen der entsprechenden Schaltfläche in der Node-RED-Benutzeroberfläche oder aber durch die Eingabe des Links http://192.168.0.126:1880/ui gestartet werden.

Die vierte Übung: Node-RED auf dem PC

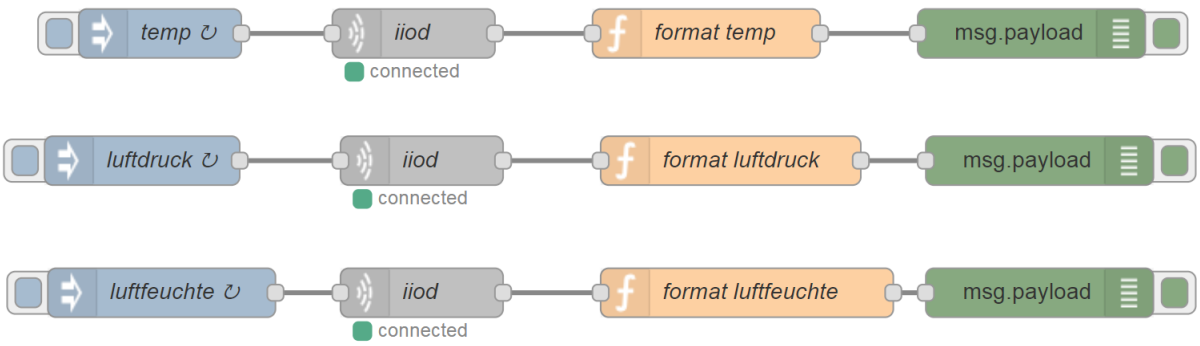
Als nächstes wollen wir mit Node-RED von einem PC aus auf den BME280-Sensor des DNP/AISS1 zugreifen. Falls Sie Node-RED noch nicht auf Ihrem PC installiert haben, holen Sie das bitte zunächst einmal nach. Für Windows-PCs finden Sie unter https://nodered.org/docs/platforms/windows eine Beschreibung der erforderlichen Installationsschritte.

Starten Sie Node-RED auf dem PC und rufen Sie den PC-Webbrowser auf. Geben Sie dann den Link http://127.0.0.1:1880 als URL ein. Dadurch wird Ihnen die Node-RED-Benutzeroberfläche Ihrer PC-Installation angezeigt.

Erzeugen Sie in der Arbeitsfläche des Node-RED-Fensters den absolut gleichen Flow, wie unter „Die zweite Übung: Erster Zugriff auf Sensordaten“ beschrieben. Ändern Sie dann lediglich in der Konfiguration des TCP Request Nodes mit dem Namen *iiod* die IP-Adresse des Servers von 127.0.0.1 auf 192.168.0.126.

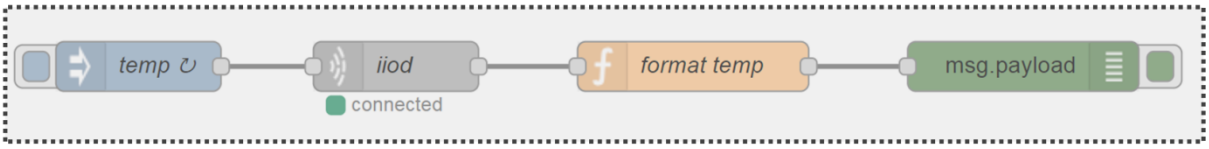
Node	Konfiguration
	

Durch zweifaches Kopieren des nun vorliegenden Flows und das anschließende Verändern der dadurch neu entstandenen Inject Nodes und Function Nodes werden wir im Rahmen dieser Übung auch die weiteren BME280-Sensorelemente für Luftdruck und relative Luftfeuchtigkeit aus-lesen. Die folgende Abbildung zeigt das erste Ziel:

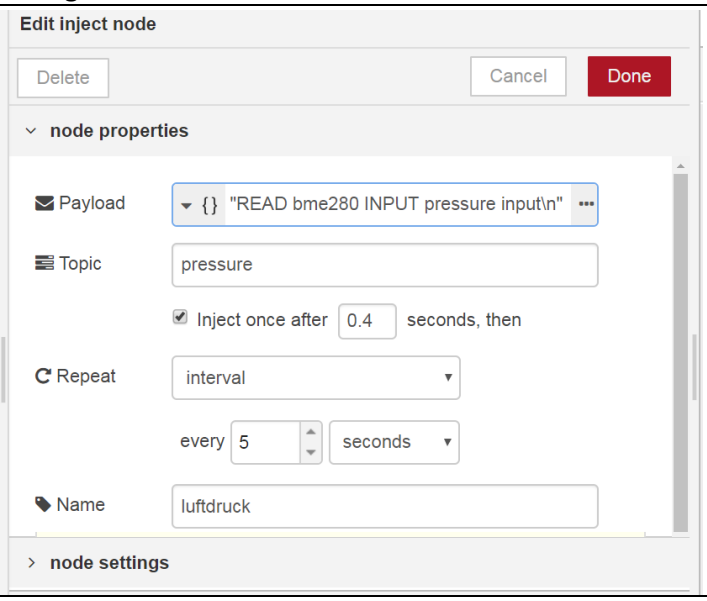
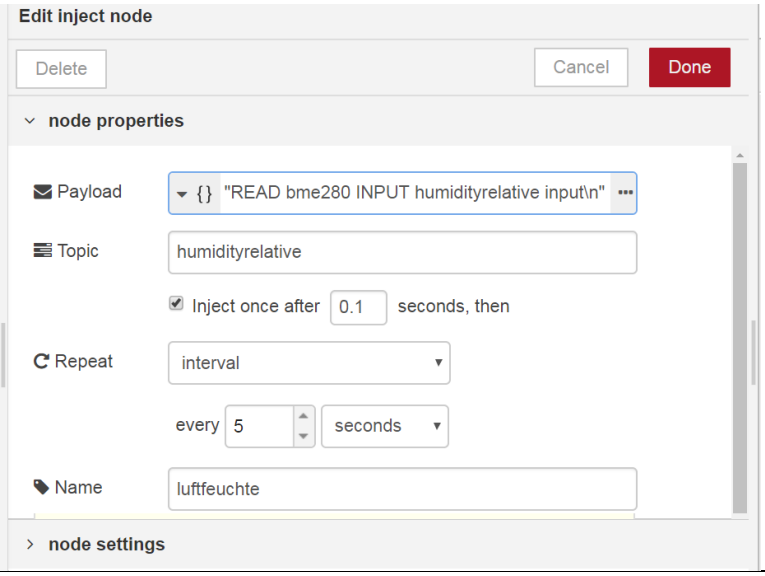


Innerhalb der Node-RED-Arbeitsfläche können Sie Nodes und ganze Flows selektieren und kopieren, ähnlich wie Textblöcke in einem Textverarbeitungsprogramm. Selektieren Sie den gesamten Flow einfach mit Hilfe der Maus (linke Maustaste drücken und Flow mit Hilfe des angezeigten Rechtsecks

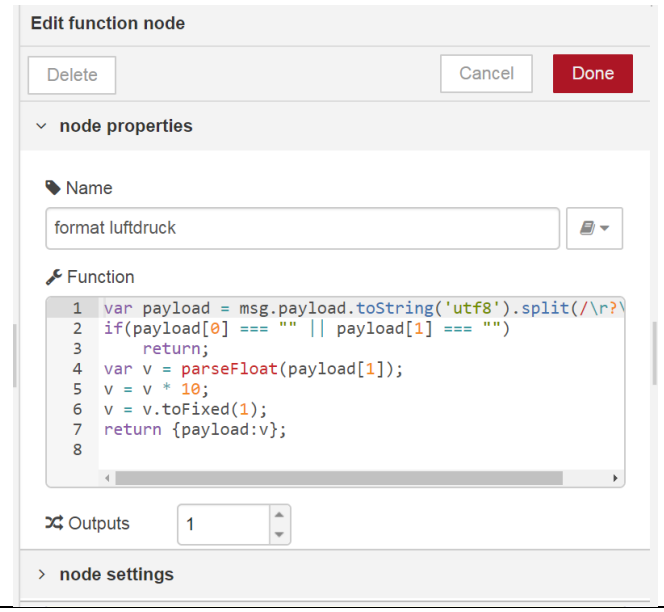
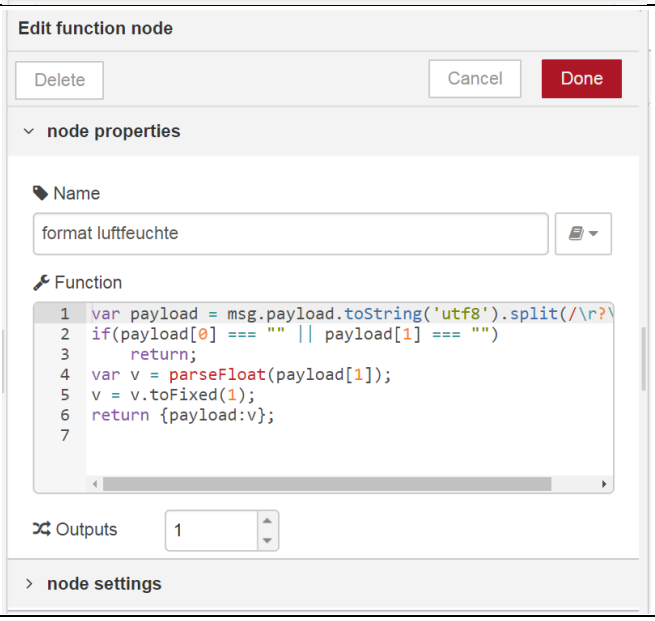
umschließen). Per CTRL-C den Flow kopieren und danach mittels CTRL-V zweimal kopieren und entsprechend in der Arbeitsfläche platzieren.



Nun müssen Sie die beiden neu hinzugekommenen Inject und Function Nodes für den Zugriff auf die BME280-Luftdruck- und Luftfeuchtesensorelemente konfigurieren. Beginnen Sie bitte mit den Inject Nodes und benutzen Sie hierfür die Angaben in der folgenden Tabelle als Vorlage.

Node	Konfiguration
	
	

Als nächstes müssen Sie nun die beiden neuen Function Nodes entsprechend konfigurieren bzw. mit geeignetem JavaScript-Code ausstatten. Die Vorlage zur Node-Konfiguration finden Sie in der hier folgenden Tabelle. Die Listings 2 und 3 enthalten den JavaScript-Code für die beiden neuen Functions Nodes mit den Namen „format luftdruck“und „format luftfeuchte“.

Node	Konfiguration
format luftdruck	
format luftfeuchte	


```
var payload = msg.payload.toString('utf8').split(/\r?\n/);
if(payload[0] === "" || payload[1] === "")
    return;
var v = parseFloat(payload[1]);
v = v * 10;
v = v.toFixed(1);
return {payload:v};
```

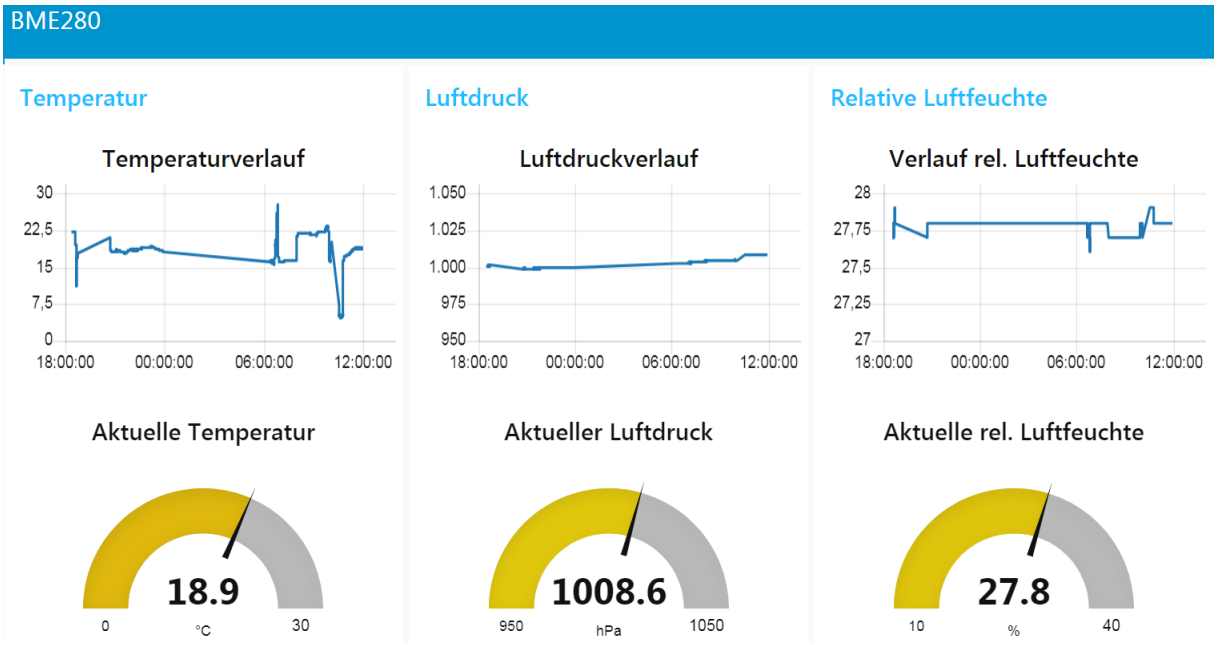
Listing 2: Quellcode für den Function Node mit dem Namen „format luftdruck“. Dieser JavaScript-Code sorgt dafür, dass die Rohdaten des BME280-Luftdrucksensorelements in eine geeignete Floating-Point-Zahl mit einer Nachkommastelle umgewandelt werden.

```
var payload = msg.payload.toString('utf8').split(/\r?\n/);
if(payload[0] === "" || payload[1] === "")
    return;
var v = parseFloat(payload[1]);
```

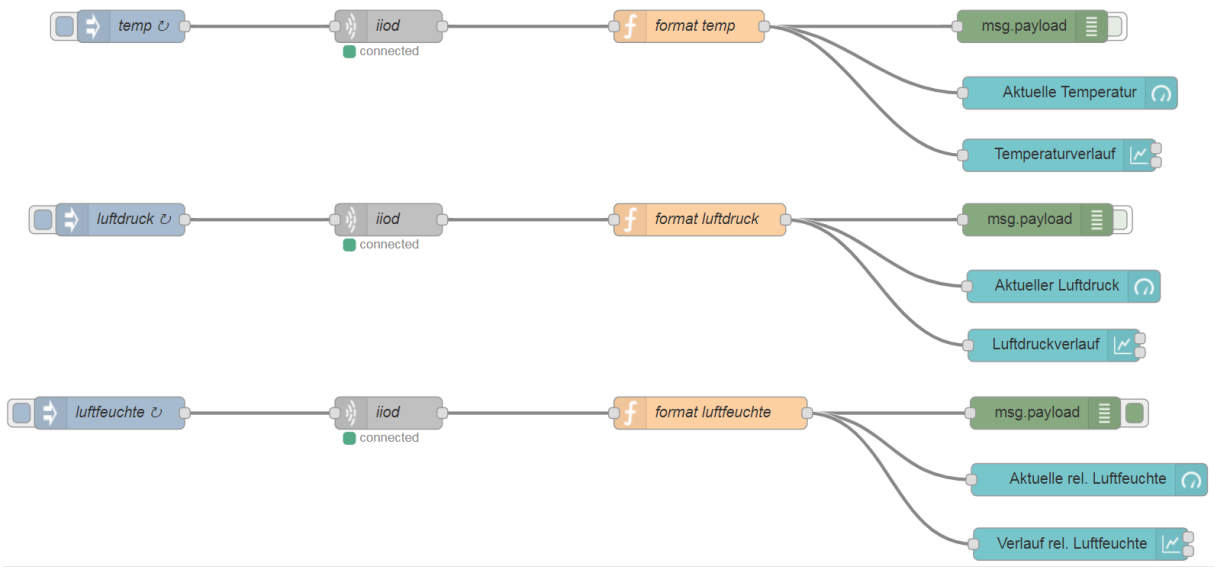
```
v = v.toFixed(1);
return {payload:v};
```

Listing 3: Quellcode für den Function Node mit dem Namen „format luftfeuchte“. Dieser JavaScript-Code sorgt dafür, dass die Rohdaten des BME280-Luftfeuchtigkeitssensorelements in eine geeignete Floating-Point-Zahl mit einer Nachkommastelle umgewandelt werden.

Sie können den Flow nun über die *Deploy*-Schaltfläche zur Ausführung bringen. Sie könnten nun auch selbstständig ein Dashboard für alle drei Sensorelemente entwerfen – siehe hierzu z. B. die folgende Abbildung:



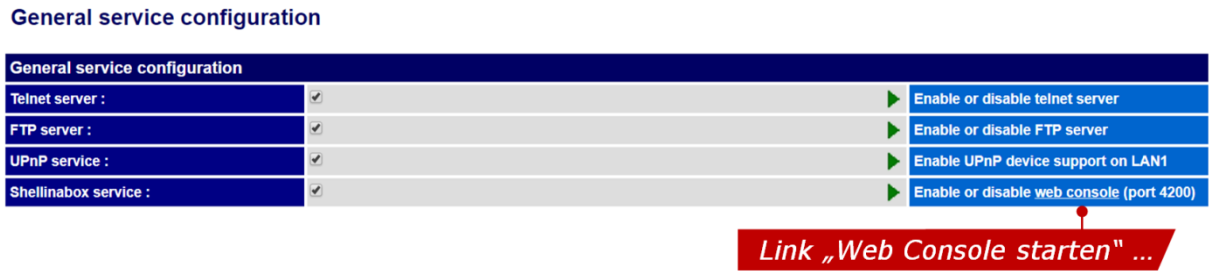
Insgesamt besteht das BME280-Dashboard aus drei Gruppen mit den Namen *Temperatur*, *Luftdruck* und *Relative Luftfeuchte*. Jede Gruppe enthält je einen *Chart Node* und *Gauge Node*.



Die fünfte Übung: Erste Schritte mit Python

Wir wollen nun mit Hilfe von Kommandozeileneingaben auf die vorinstallierte Python 3-Laufzeitumgebung zugreifen und ein erstes selbstentwickeltes Python-Skript ausführen.

Rufen Sie per Webbrowser über den Link http://192.168.0.126:7777 die Web-basierte Benutzeroberfläche des DNP/AISS1 auf. Melden Sie sich dann bitte mit einem gültigen Benutzernamen und dem dazu gehörenden Passwort an. Wählen Sie nach der erfolgreichen Anmeldung in der Weboberfläche bitte im Menüpunkt *Services* den Unterpunkt *General* aus. Klicken Sie danach auf den Link für die *Web Console*. Dadurch wird in einem weiteren Browser-Fenster der *Shell-In-a-Box Service* gestartet.

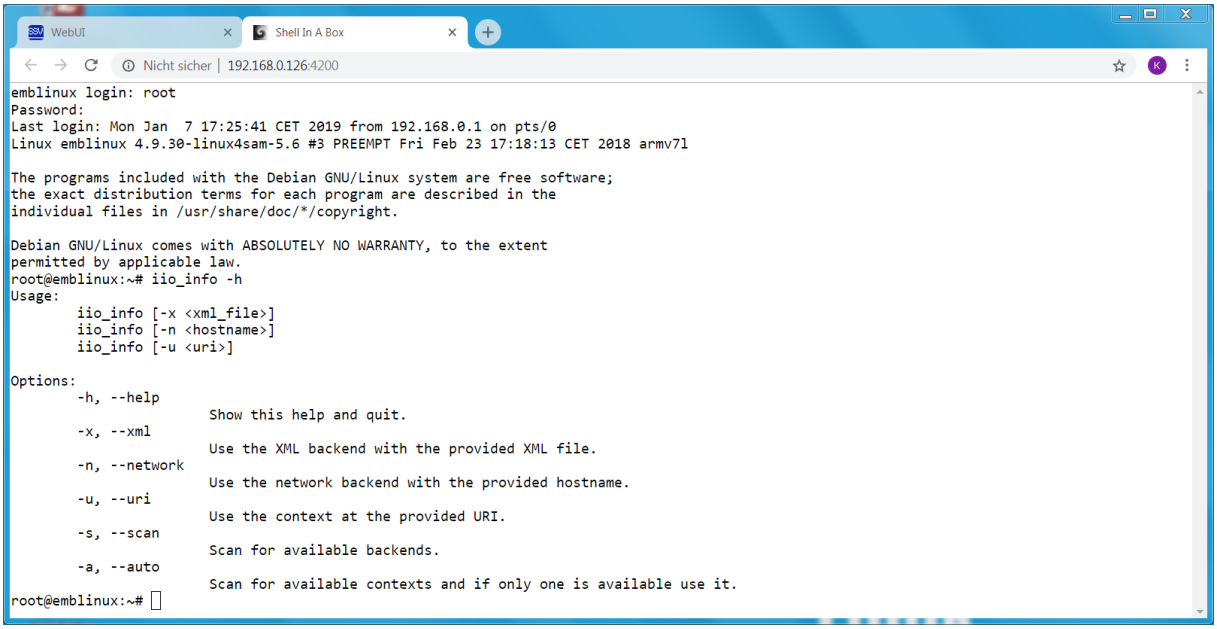


Melden Sie sich dann zunächst mit einem gültigen Benutzernamen und dem dazu gehörenden Passwort in der Web Console des Shell-In-a-Box Service an.

Eingabe	Funktion
`iio_info -h`	Übersicht zum Industrial I/O-Treiber-Hilfsprogramm „iio_info“ ausgeben.
`cd /tmp`	In das Verzeichnis mit dem Namen „tmp“ wechseln.
`ls -al`	Dateinamen im aktuellen Verzeichnis anzeigen.

Tabelle 3: Beispiele für Kommandozeileneingaben in der Web Console

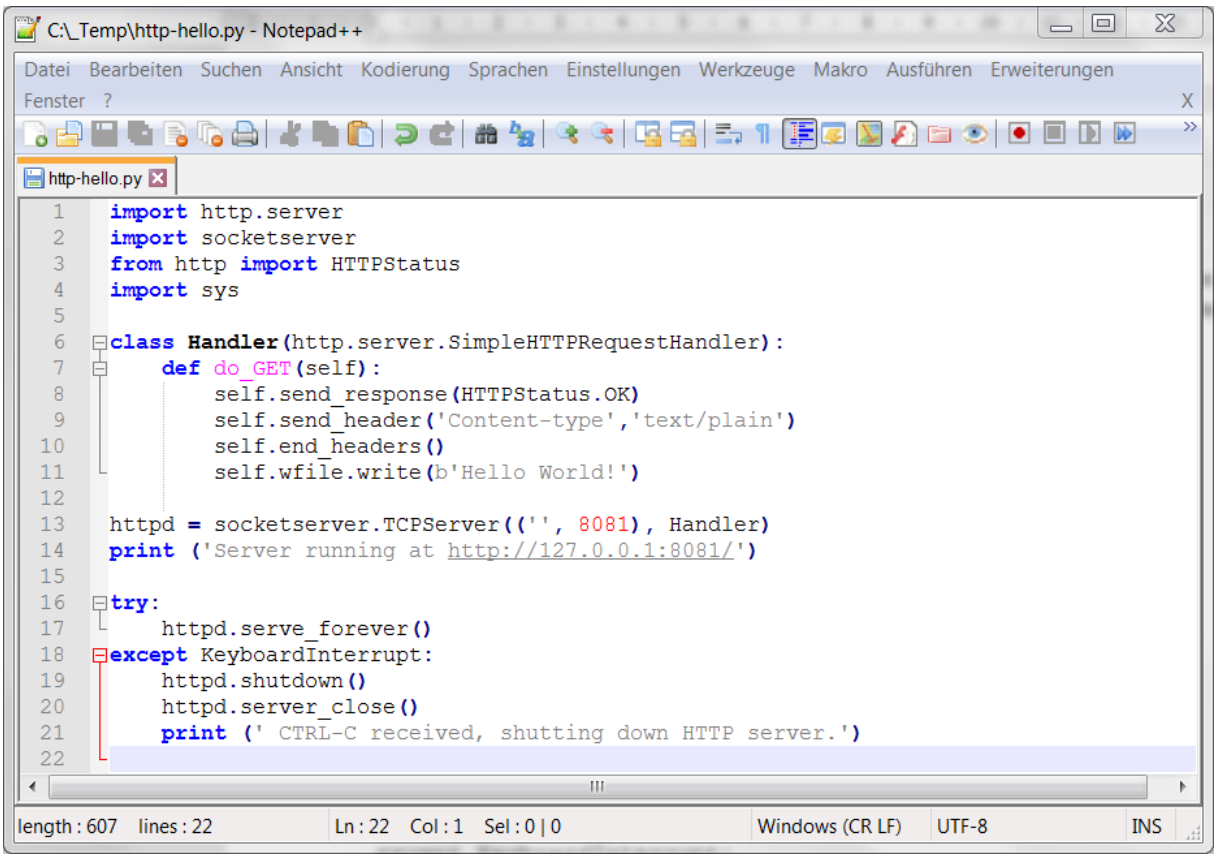
Probieren Sie nun die Kommandozeilenbeispiele aus der Tabelle 3 aus. Beachten Sie bitte, dass jede Kommandozeileneingabe mit der Eingabe-Taste abgeschlossen wird.



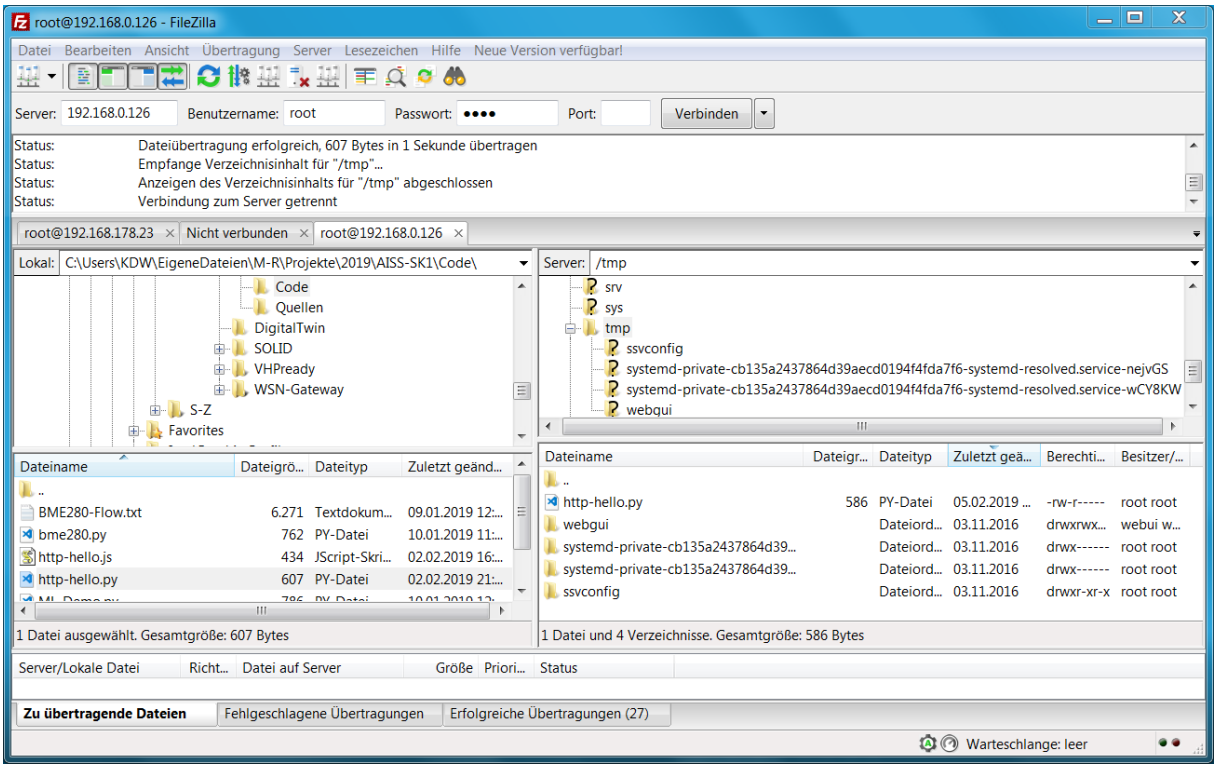
Starten Sie auf Ihrem PC nun einen Editor zum Bearbeiten von Quellcodes. Übertragen Sie den hier folgenden Python-Quellcode in eine neue Datei mit dem Namen *http-hello.py*. Speichern Sie diese Datei in einem Verzeichnis Ihrer Wahl auf dem PC.

```
import http.server
import socketserver
from http import HTTPStatus
import sys

class Handler(http.server.SimpleHTTPRequestHandler):
    def do_GET(self):
        self.send_response(HTTPStatus.OK)
        self.send_header('Content-type','text/plain')
        self.end_headers()
        self.wfile.write(b'Hello World!')

httpd = socketserver.TCPServer(('', 8081), Handler)
print ('Server running at http://127.0.0.1:8081/')

try:
    httpd.serve_forever()
except KeyboardInterrupt:
    httpd.shutdown()
    httpd.server_close()
    print (' CTRL-C received, shutting down HTTP server.')
```

Listing 4: Python-Quellcode für einen einfachen HTTP-Server (Webserver)



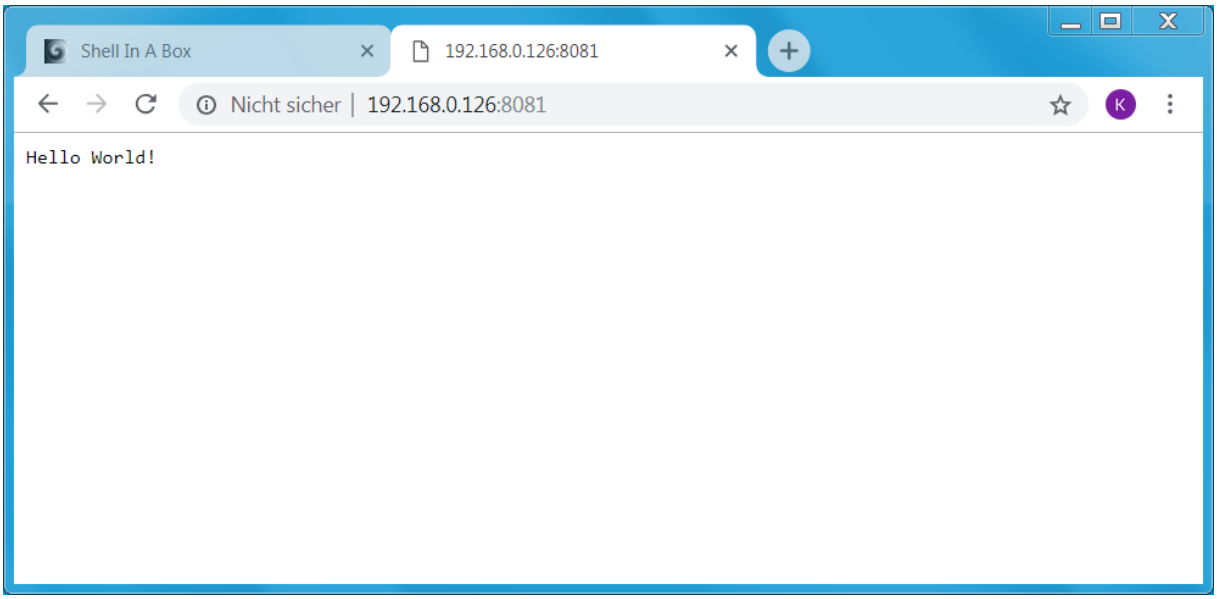
Übertragen Sie die Datei *http-hello.py* mit Hilfe eines FTP-Clients vom PC aus in das Verzeichnis */tmp* des DNP/AISS1. **Achtung:** Beachten Sie bitte, dass es sich bei */tmp* um ein RAM-Disk-Verzeichnis handelt. Das heißt, alle hier gespeicherten Dateien gehen beim nächsten Ausschalten des DNP/AISS1 verloren.



Geben Sie nun in der Web Console des Shell-In-a-Box Service das Kommando *cd /tmp* ein und starten Sie dann im Verzeichnis */tmp* mit der Eingabe *python3 http-hello.py* den HTTP-Server aus dem Listing 4.

Der HTTP-Server auf dem DNP/AISS1 ist nun betriebsbereit und wartet auf Anfragen. Öffnen Sie ein weiteres Fenster und geben Sie den folgenden Link als Adresse ein: http://192.168.0.126:8081

Achtung: Bitte unbedingt die TCP-Port-Nummer 8081 im Link beachten. Unter diesem Port wartet der HTTP-Server aus dem Listing 4 auf die Anfragen.



Geben Sie bitte CTRL-C in der Web Console des Shell-In-a-Box Service ein, um den HTTP-Server aus dem Listing 4 wieder zu beenden.

Die sechste Übung: Erste Schritte mit Node.js

Wir wollen nun per Kommandozeile auf die vorinstallierte Node.js-JavaScript-Laufzeitumgebung des DNP/AISS1 zugreifen und einen ersten selbstentwickelten JavaScript-Code ausführen.

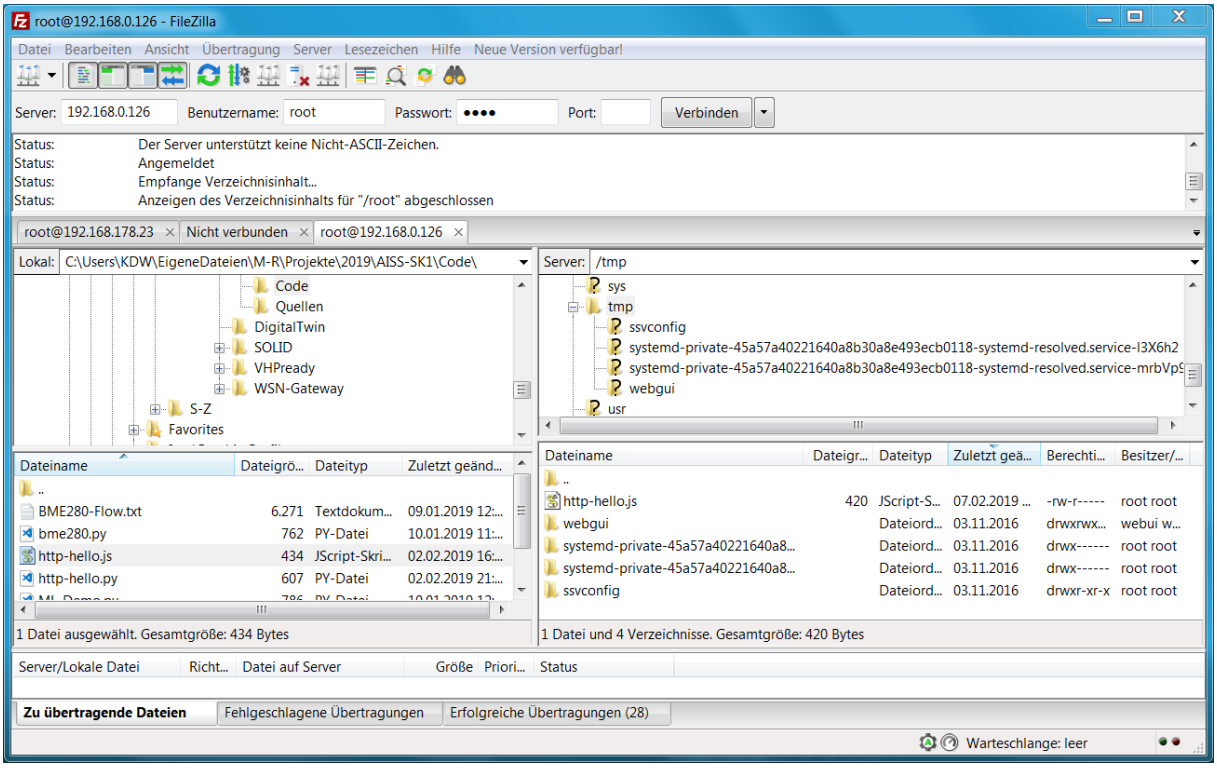
Achtung: Bei dieser Übung wird vorausgesetzt, dass Sie zuvor „Die fünfte Übung: Erste Schritte mit Python“ erfolgreich durchgeführt haben.

Starten Sie auf Ihrem PC nun einen Editor zum Bearbeiten von Quellcodes. Übertragen Sie folgenden JavaScript-Quellcode in eine neue Datei mit dem Namen *http-hello.js*. Speichern Sie diese Datei auf dem PC in einem Verzeichnis Ihrer Wahl.

```
var http = require("http");

http.createServer(function (request, response) {
   response.writeHead(200, {'Content-Type': 'text/plain'});
   response.end('Hello World!\n');
}).listen(8081);

console.log('Server running at http://127.0.0.1:8081/');
```

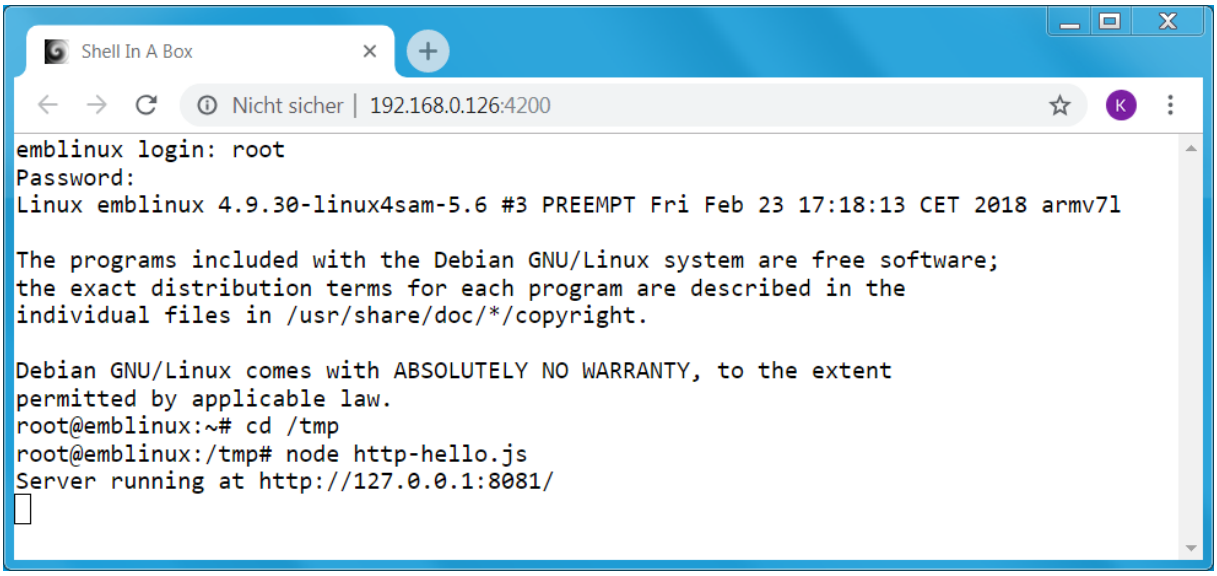
Listing 5: JavaScript-Quellcode für einen einfachen HTTP-Server (Webserver)

Übertragen Sie die Datei *http-hello.js* mit Hilfe eines FTP-Clients vom PC aus in das Verzeichnis */tmp* des DNP/AISS1.

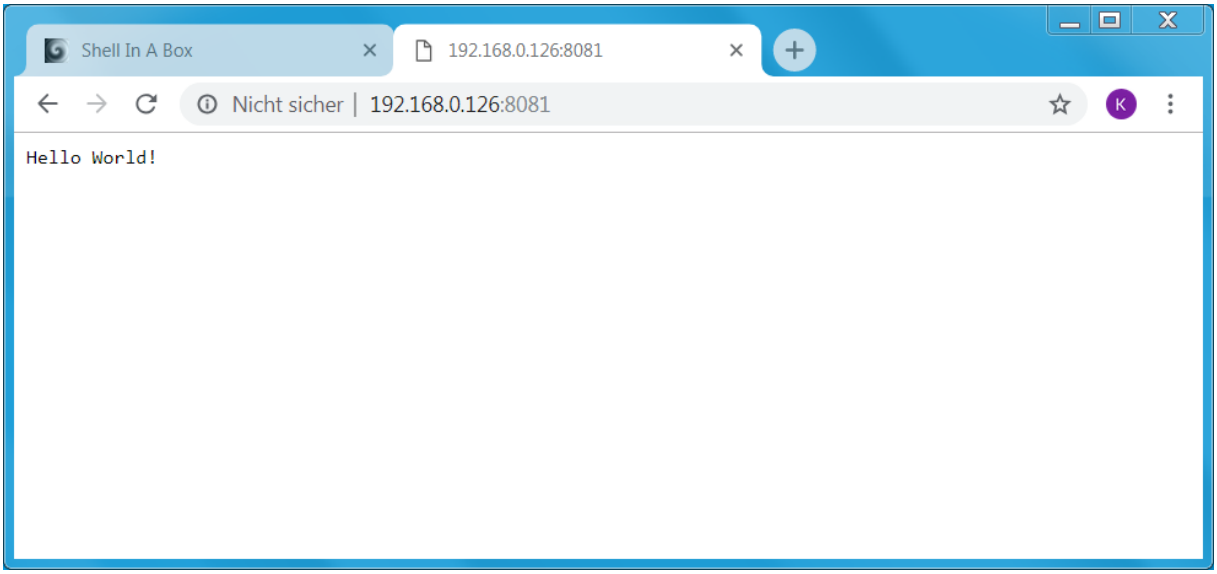


Wählen Sie in der Weboberfläche bitte im Menüpunkt *Services* den Unterpunkt *General* aus. Klicken Sie danach auf den Link für die *Web Console*, um diese zu starten. Melden Sie sich dann zunächst mit einem gültigen Benutzernamen und dem dazu gehörenden Passwort in der Web Console des Shell-In-a-Box Service an.

Geben Sie nun in der Web Console des Shell-In-a-Box Service das Kommando *cd /tmp* ein und starten Sie dann im Verzeichnis */tmp* mit der Eingabe *node http-hello.js* den HTTP-Server aus dem Listing 5.



Der HTTP-Server auf dem DNP/AISS1 ist nun betriebsbereit und wartet auf die Anfragen eines HTTP-Clients. Öffnen Sie ein weiteres Browser-Fenster und geben Sie den folgenden Link als Adresse ein: http://192.168.0.126:8081

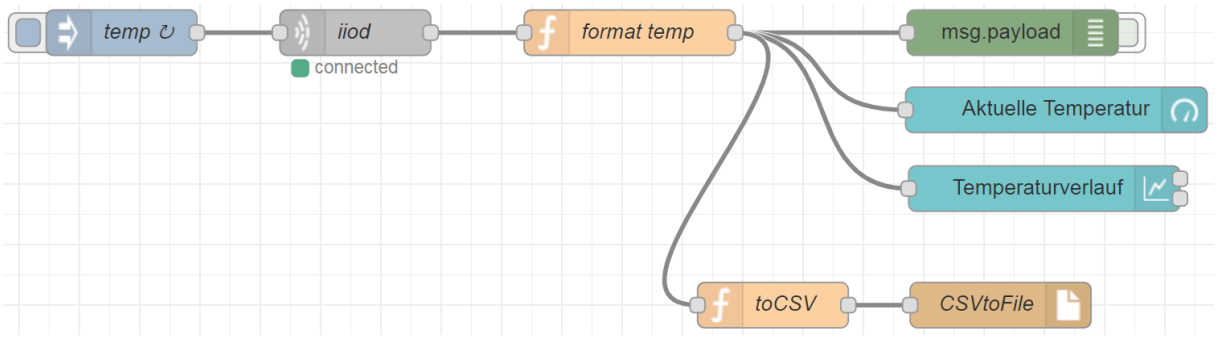


Geben Sie bitte CTRL-C in der Web Console des Shell-In-a-Box Service ein, um den JavaScript-HTTP-Server und Node.js wieder zu beenden.

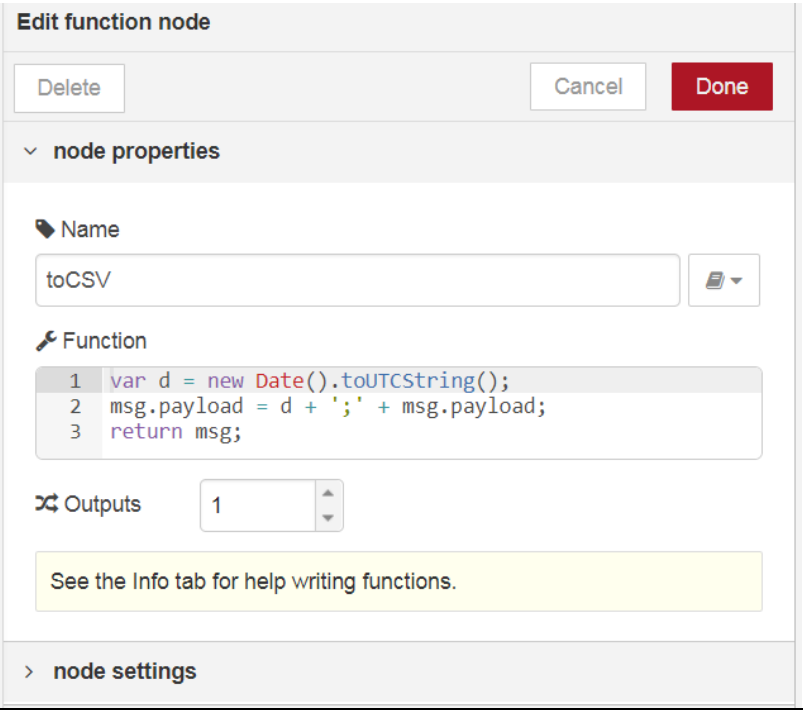
Die siebte Übung: Einfacher Temperatur-Datenlogger

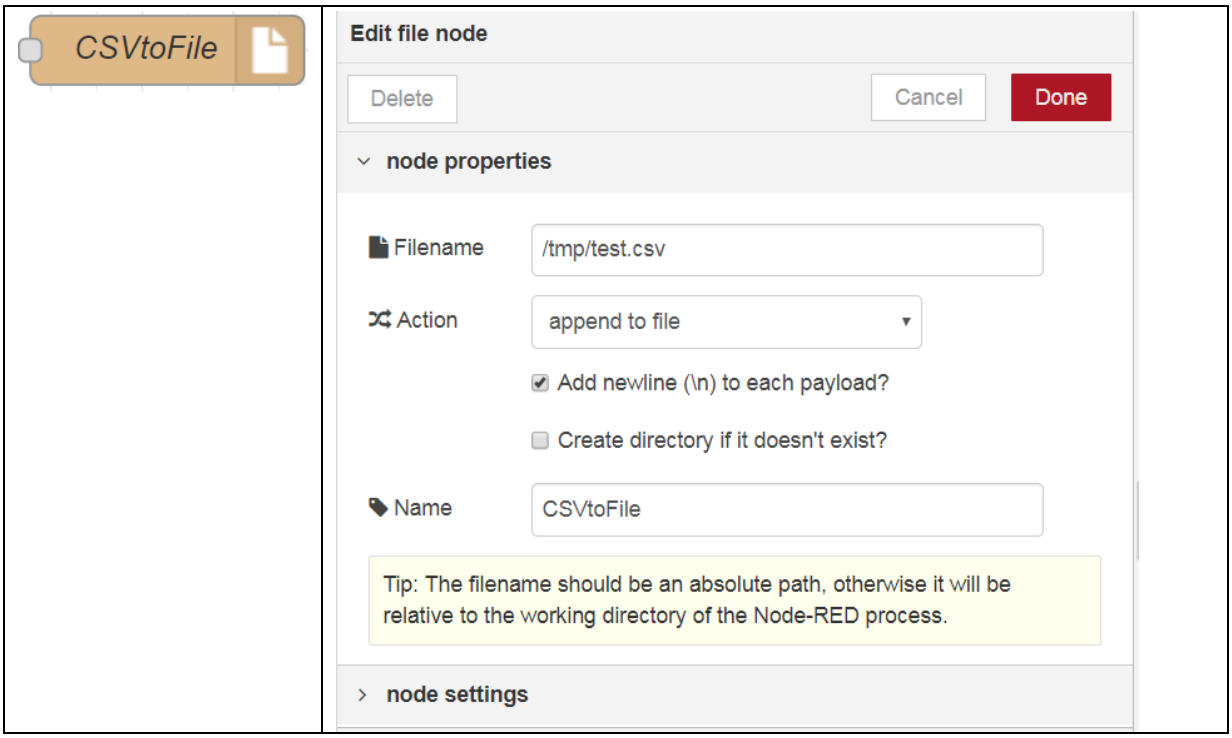
Wir wollen nun den Temperaturmesswert des BME280-Sensors periodisch mit einem Zeitstempel versehen und in einer Datei */tmp/test.csv* speichern. Diese Datei können wir dann per FTP zum PC übertragen und dort analysieren. Als Ausgangspunkt für diese Übung nutzen wir die Node-RED-Beispielkonfiguration aus „Die dritte Übung: Hinzufügen eines Dashboards“.

Um unser Ziel zu erreichen, erweiterten wir im ersten Schritt den Temperatur-Teil des Node-RED-Flows auf dem DNP/AISS1 um jeweils einen *Function Node* und *File Node*. Die folgende Abbildung zeigt die angestrebte Konfiguration.

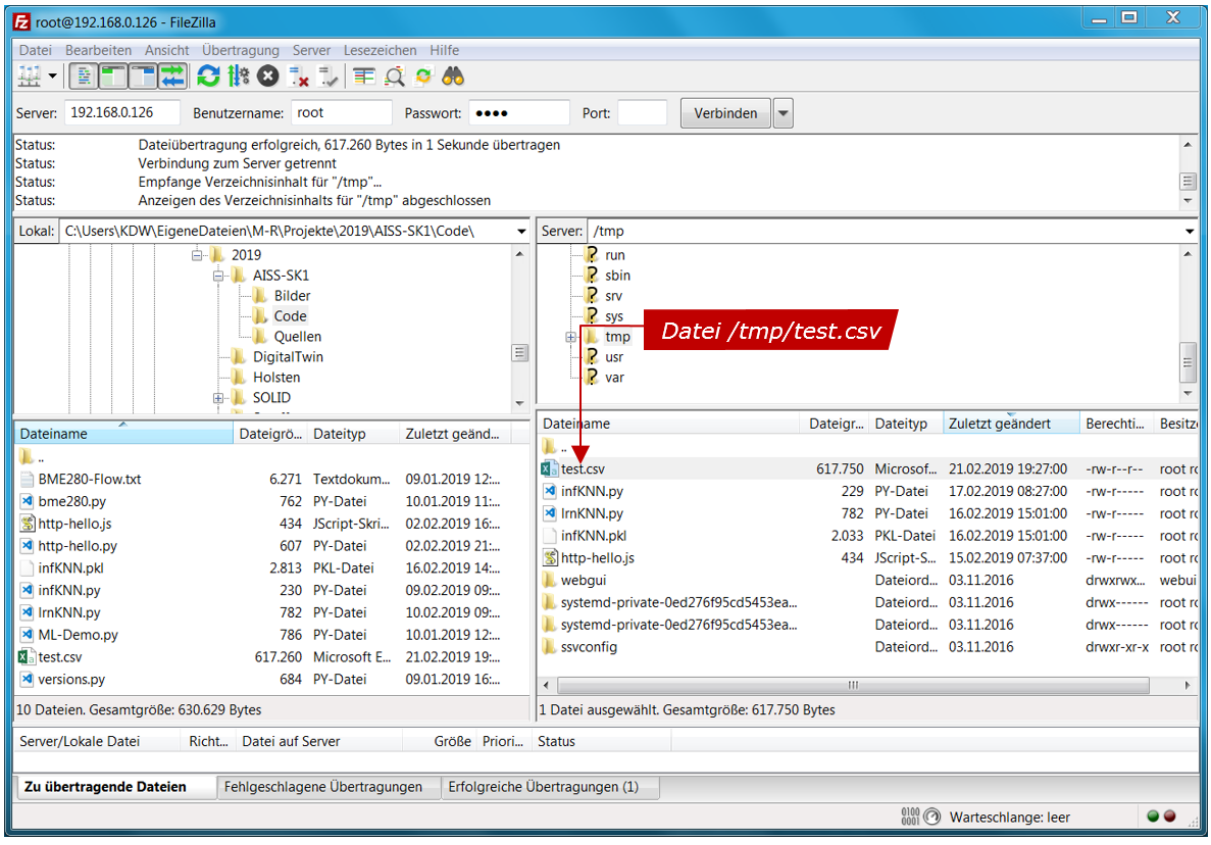


Ziehen Sie bitte die beiden zusätzlich benötigten Nodes in den Arbeitsbereich und verdrahten Sie die Nodes wird in der Abbildung dargestellt. Konfigurieren Sie die beiden neuen Nodes mit den Vorgaben aus der folgenden Tabelle und betätigen Sie dann die Node-RED-Deploy-Schaltfläche.

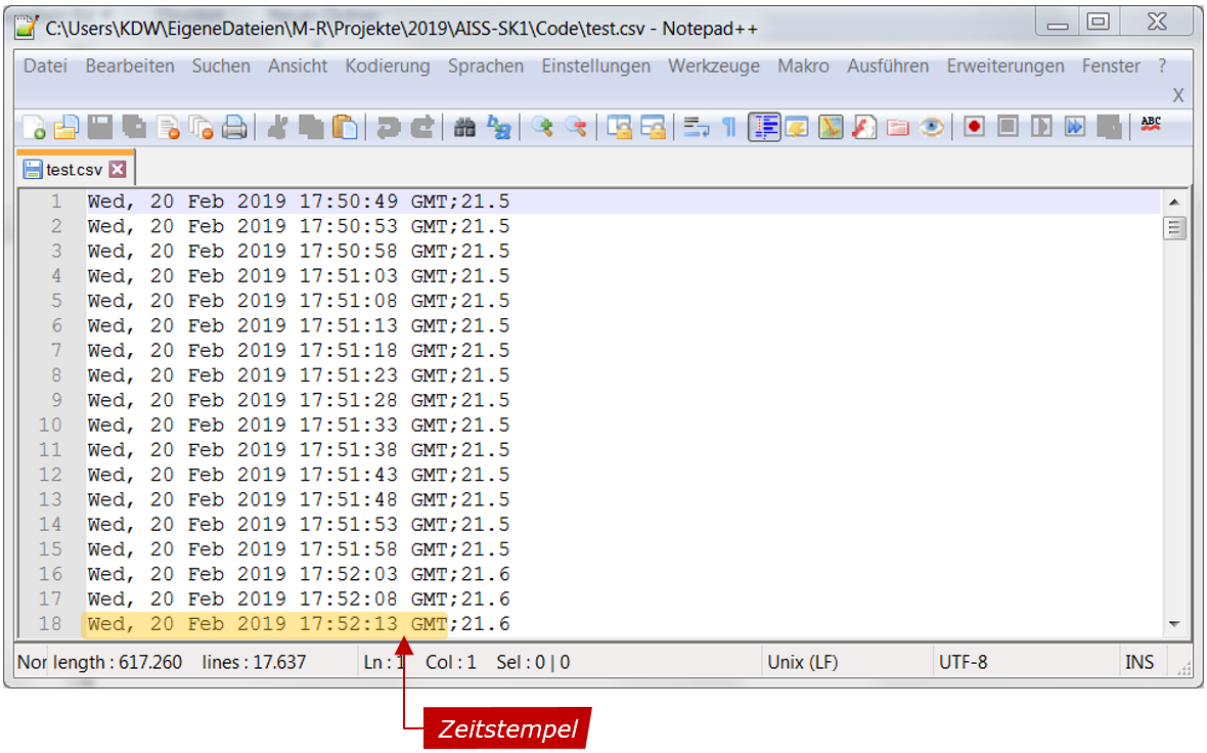
Node	Konfiguration
	



Wenn Sie nun den neuen Node-RED-Flow eine Zeitlang auf dem DNP/AISS1 laufen lassen, entsteht im Verzeichnis */tmp* die CSV-Datei *test.csv* (CSV = Comma Separated Values, gängiges Textdateiformat zum Austausch einfach strukturierter Daten). Mit der bisherigen Konfiguration wird alle 5 Sekunden jeweils ein neuer Temperaturwert in diese CSV-Datei geschrieben. Verantwortlich dafür ist der *Inject Node* ganz links. Der wurde auf eine Intervallzeit von 5 Sekunden konfiguriert. **Achtung:** Beachten Sie bitte, dass es sich bei */tmp* um ein RAM-Disk-Verzeichnis handelt. Das heißt, alle hier gespeicherten Dateien und Daten gehen beim nächsten Ausschalten des DNP/AISS1 verloren.



Warten Sie auf jeden Fall etwas ab, bis sich einige Temperaturdaten in der Datei *test.csv* gesammelt haben. Übertragen Sie dann diese Datei mit Hilfe eines FTP-Clients aus dem DNP/AISS1-Verzeichnis */tmp* in ein Verzeichnis Ihrer Wahl auf Ihrem PC.



Beachten Sie bitte folgendes: Jede einzelne Zeile der Datei *test.csv* besteht aus zwei Elementen, die jeweils durch ein ‚;'-Zeichen (Semikolon) voneinander getrennt sind:

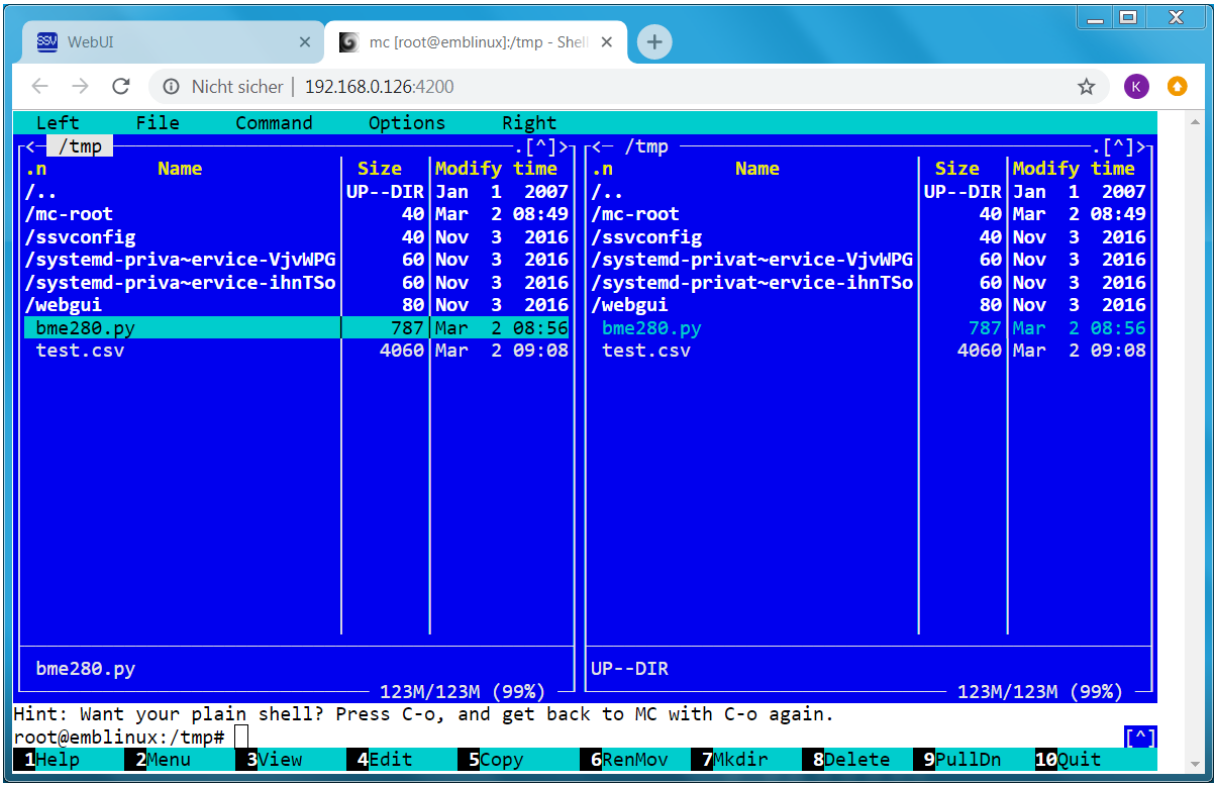
1. Dem Zeitstempel, der im Function Node mit dem Namen "toCSV" hinzugefügt wird.
2. Dem aktuellen Temperaturmesswert aus dem BME280-Sensor des DNP/AISS1.

Sie können *test.csv* auf dem PC nun genauer untersuchen oder den Inhalt mit Programmen weiterverarbeiten, die Dateien im CSV-Format importieren können.

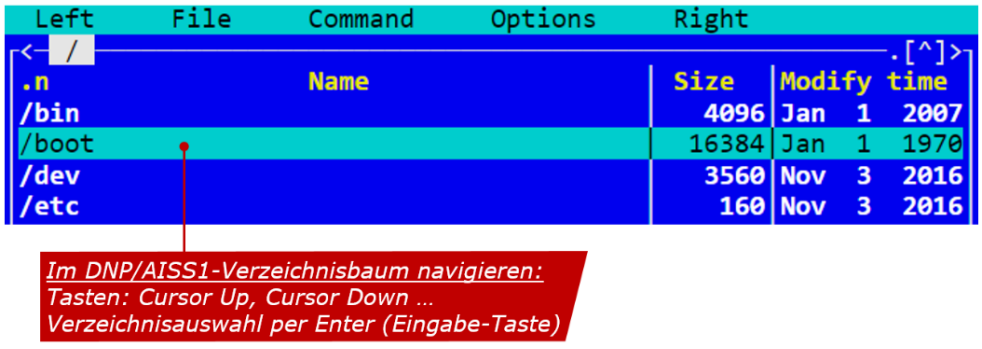
Die achte Übung: Dateien auf dem DNP/AISS1 editieren

Wenn Sie Python-, Node.js-JavaScript-Quelltexte und andere Textdateien direkt auf dem DNP/AISS1 per Webbrowser anschauen oder verändern wollen, können Sie dafür den s.g. *Midnight Commander* benutzen. (siehe hierzu auch https://de.wikipedia.org/wiki/Midnight_Commander).

Wählen Sie in der Weboberfläche bitte im Menüpunkt *Services* den Unterpunkt *General* aus. Klicken Sie danach auf den Link für die *Web Console*, um diese zu starten. Melden Sie sich dann zunächst mit einem gültigen Benutzernamen und dem dazu gehörenden Passwort in der Web Console des Shell-In-a-Box Service an. Geben Sie nun in der Web Console des Shell-In-a-Box Service das Kommando *mc* ein. Dadurch wird der *Midnight Commander* gestartet.

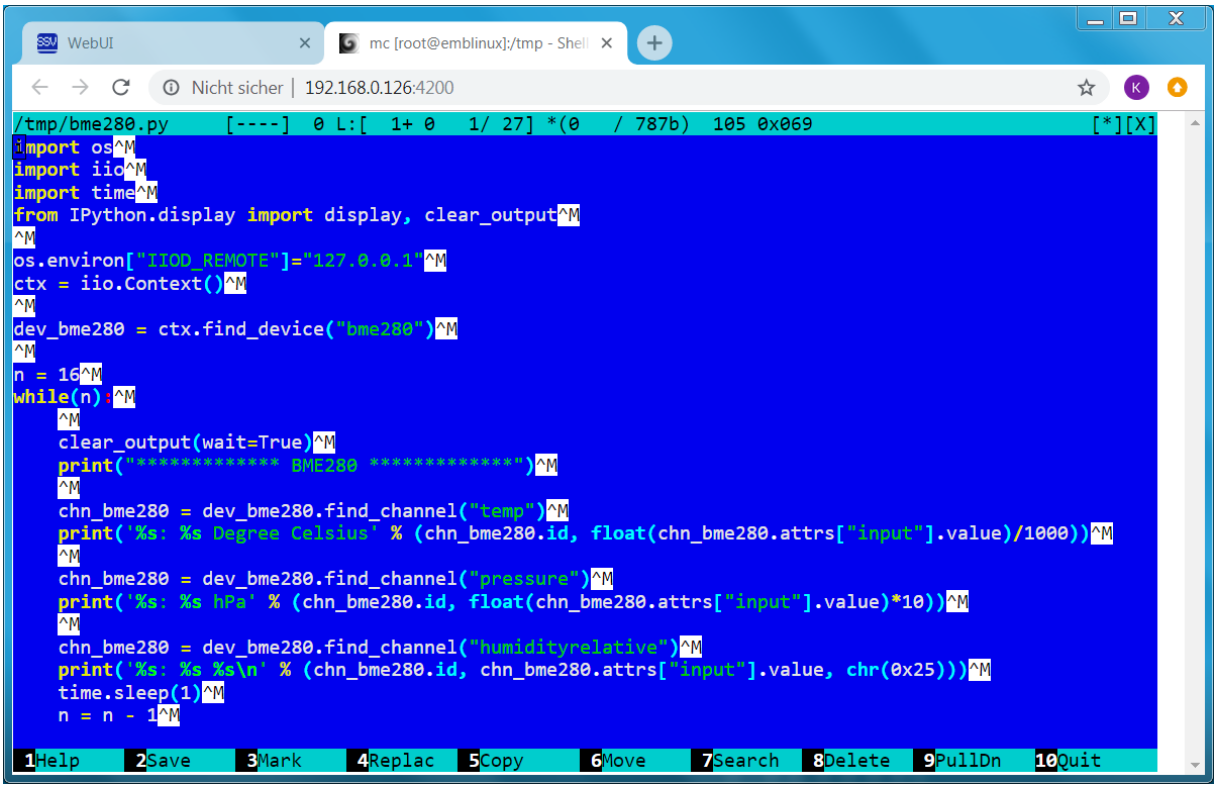


Über die Oberfläche dieses Dateimanagers können Sie im Verzeichnisbaum des DNP/AISS1-Linux-Betriebssystems navigieren, Dateien öffnen, um den Inhalt anzuschauen und – unter der Voraussetzung, dass sich eine Datei in einem Schreib/Lese-Verzeichnis (also in einem RAM-Disk-Verzeichnis) befindet – den Inhalt einer Datei verändern.

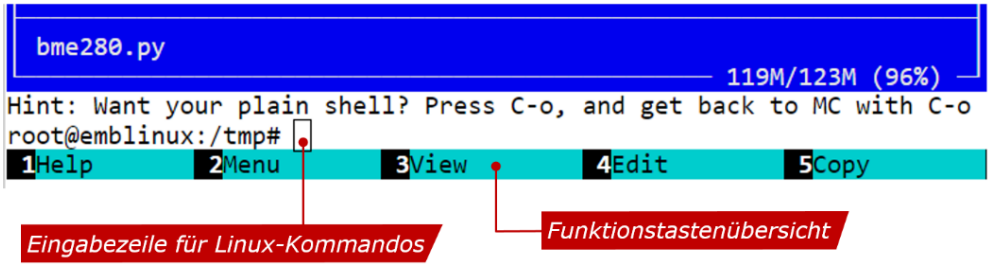


Wenn der horizontale Auswahl-Cursor des Midnight Commander über einer Textdatei steht, können Sie diese Datei zum Beispiel über die F3-Taste (Funktionstaste) Ihrer PC-Tastatur zur Anzeige bringen.

Falls Sie beispielsweise eine Datei mit einem Python-Quellcode verändern (editieren) wollen, positionieren Sie bitte zunächst den horizontalen Auswahl-Cursor über den jeweiligen Dateinamen und betätigen Sie dann die Funktionstaste F4.



Die Oberfläche des Midnight Commander bietet Ihnen in der unteren Textzeile stets eine Übersicht der jeweils möglichen Funktionstasteneingaben und den dazu gehörenden Funktionen. Die jeweilige Zuordnung ist kontextabhängig.

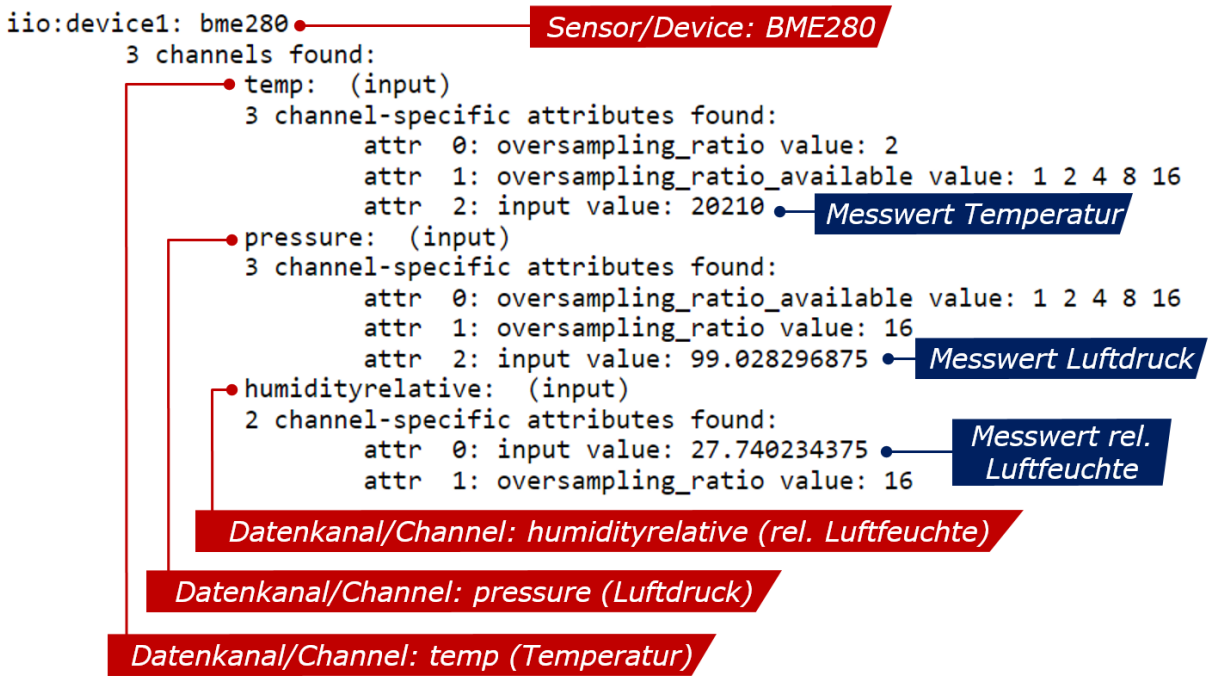


Im Navigationsmodus besteht direkt über der unteren Textzeile die Möglichkeit der direkten Eingabe von Linux-Kommandozeilen, die dann vom DNP/AISS1-Betriebssystem ausgeführt werden.

Sie können dort z. B. durch die Eingabe *python3 http-hello.py* einen Python-Quellcode erneut ausführen lassen, den Sie zuvor mit Hilfe der Editorfunktion des Midnight Commander verändert haben.

Die neunte Übung: CSV-Datei mit BME280-Messdaten erzeugen

Die drei Bosch-Sensoren des DNP/AISS1 sind per I2C und dem IIO Framework *libiio* (Analog Devices LibIIO) mit dem Linux-Betriebssystem des DNP/9535 Microrechner verbunden. Aus LibIIO-Sicht ist jeder einzelne Sensor eine Device mit verschiedenen Datenkanälen (Channels). Die hier folgende Abbildung zeigt den Zusammenhang für den BME280 Environment Sensor:



Die BME280-Sensorelemente für Temperatur, Luftdruck und rel. Luftfeuchte bilden die jeweilen Channels. Jeder einzelne Channels besteht aus verschiedenen Attributen. Die aktuellen Messwerte der Sensorelemente sind in den Attributen zu finden.

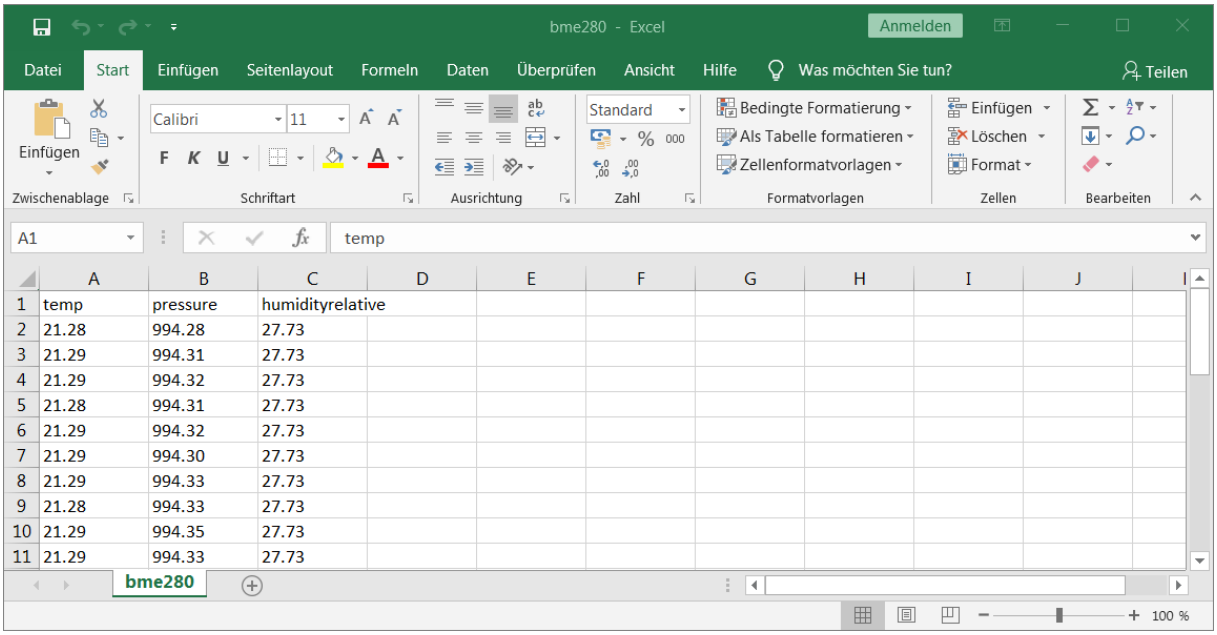
Das hier folgende Listing zeigt ein vollständiges BME280-Zugriffsbeispiel. Die einzelnen Sensorelemente werden einmal pro Minute insgesamt 32-mal ausgelesen. Die Messwerte werden in dem Beispiel als Fließkommazahlen mit zwei Nachkommastellen aufbereitet. Die jeweils gemessene Temperatur, Luftdruck und rel. Luftfeuchte wird in einer Textzeile zusammengefasst und in einer CSV-Datei mit dem Namen *bme280.csv* gespeichert.

```
import os
import iio
import time

os.environ["IIOD_REMOTE"]="127.0.0.1"
ctx = iio.Context()

dev_bme280 = ctx.find_device("bme280")

n = 32
file = open("bme280.csv", "w")
file.write("temp;" + "pressure;" + "humidityrelative" + '\n')
file.close()
while (n):
   chn_bme280 = dev_bme280.find_channel("temp")
   t = float(chn_bme280.attrs["input"].value)/1000
```

```
str_t = "%.2f" % t

chn_bme280 = dev_bme280.find_channel("pressure")
p = float(chn_bme280.attrs["input"].value)*10
str_p = "%.2f" % p

chn_bme280 = dev_bme280.find_channel("humidityrelative")
h = float(chn_bme280.attrs["input"].value)
str_h = "%.2f" % h
print (str_t + ';' + str_p + ';' + str_h)
file = open("bme280.csv", "a")
file.write(str_t + ';' + str_p + ';' + str_h + '\n')
file.close()
n = n - 1
time.sleep(1)
```

Listing 6: Python-Quellcode, um die drei BME280-Sensorelemente für Temperatur, Luftdruck und rel. Luftfeuchte jeweils einmal pro Minute auszulesen und das Ergebnis in einer CSV-Datei zu speichern

Übertragen Sie den Python-Quellcode aus dem Listing 6 in eine Datei. Speichern Sie diese danach im DNP/AISS1-Verzeichnis */tmp* und bringen Sie die Quellcodedatei dann zur Ausführung. Benutzen Sie „Die fünfte Übung: Erste Schritte mit Python“ als Vorlage für Ihre Vorgehensweise.



	A	B	C
1	temp	pressure	humidityrelative
2	21.28	994.28	27.73
3	21.29	994.31	27.73
4	21.29	994.32	27.73
5	21.28	994.31	27.73
6	21.29	994.32	27.73
7	21.29	994.30	27.73
8	21.29	994.33	27.73
9	21.28	994.33	27.73
10	21.29	994.35	27.73
11	21.29	994.33	27.73

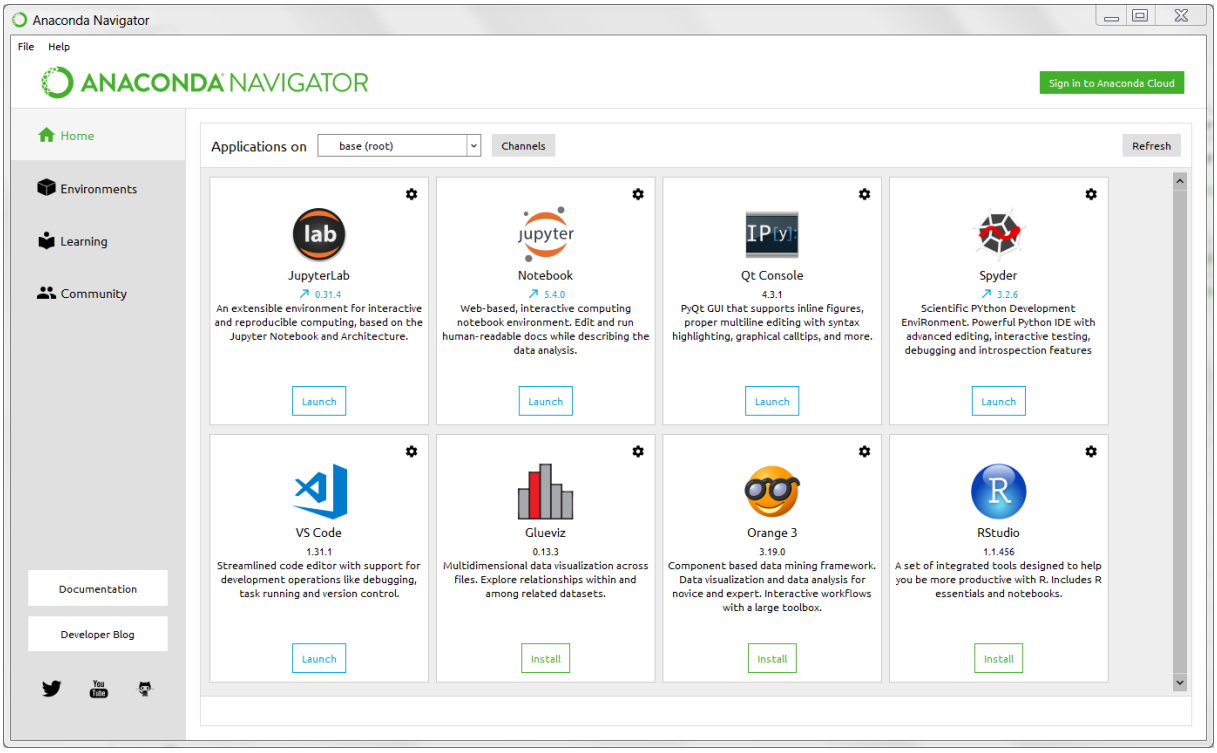
Beachten Sie bitte, dass Sie die Datei *bme280.csv* bei Bedarf per FTP von DNP/AISS1 auf Ihren PC übertragen können und dort mit Excel betrachten bzw. weiterverarbeiten können.

Beachten Sie weiterhin bitte die Codezeilen 10 und 31 im Listing 6. Über Zeile 10 (n = 32) bestimmen Sie, wieviel Messungen insgesamt durchgeführt und in der Datei *bme280.csv* gespeichert werden. Mit Hilfe von Zeile 31 (time.sleep(1)) legen Sie die Zeit in Sekunden fest, die zwischen zwei einzelnen Messungen abgewartet wird. Wenn Sie beispielsweise eine Messung pro Minute in der CSV-Datei speichern möchten, müsste hier *time.sleep(60)* stehen.

Die zehnte Übung: CSV-Datei mit PC-Entwicklungsumgebung auswerten

Wir wollen nun die CSV-Datei aus „Die neunte Übung: CSV-Datei mit BME280-Messdaten erzeugen“ mit Hilfe einer Data-Science-Entwicklungsumgebung auf einem PC genauer untersuchen. Dabei werden wir die Messwerte aus der CSV-Datei *bme280.csv* in Liniendiagrammen, Boxplots, Histogrammen usw. visualisieren.

Diese Übung erfordert eine lauffähige Version der Open-Source-Data-Science-Entwicklungsumgebung *Anaconda* auf Ihrem PC. Für Windows-PCs, Linux und MacOS finden Sie im Internet unter dem Link https://www.anaconda.com/distribution/#windows eine Download-Möglichkeit und die Beschreibung der erforderlichen Installationsschritte. **Achtung:** Bitte beim Download beachten, dass wir die Anaconda-Version für Python 3.x benötigen.



Über den *Anaconda Navigator* als Benutzeroberfläche werden wir u.a. *Jupyter Notebook* nutzen. Das ist eine weit verbreitete Open-Source-Programmierumgebung, um per Webbrowser interaktive Dokumente mit Python-Code zu erzeugen, weiterzugeben und zu bearbeiten. Siehe hierzu auch https://jupyter.org/.

Klicken Sie nun bitte in der Anaconda Navigator-Benutzeroberfläche innerhalb der Kachel des Jupyter Notebook auf die *Launch*-Schaltfläche. Warten Sie dann, bis in Ihrem Webbrowserfenster der Jupyter Notebook-Eingangsdialog zum Navigieren in den PC-Verzeichnissen und zum Öffnen vorhandener Projekte angezeigt wird.

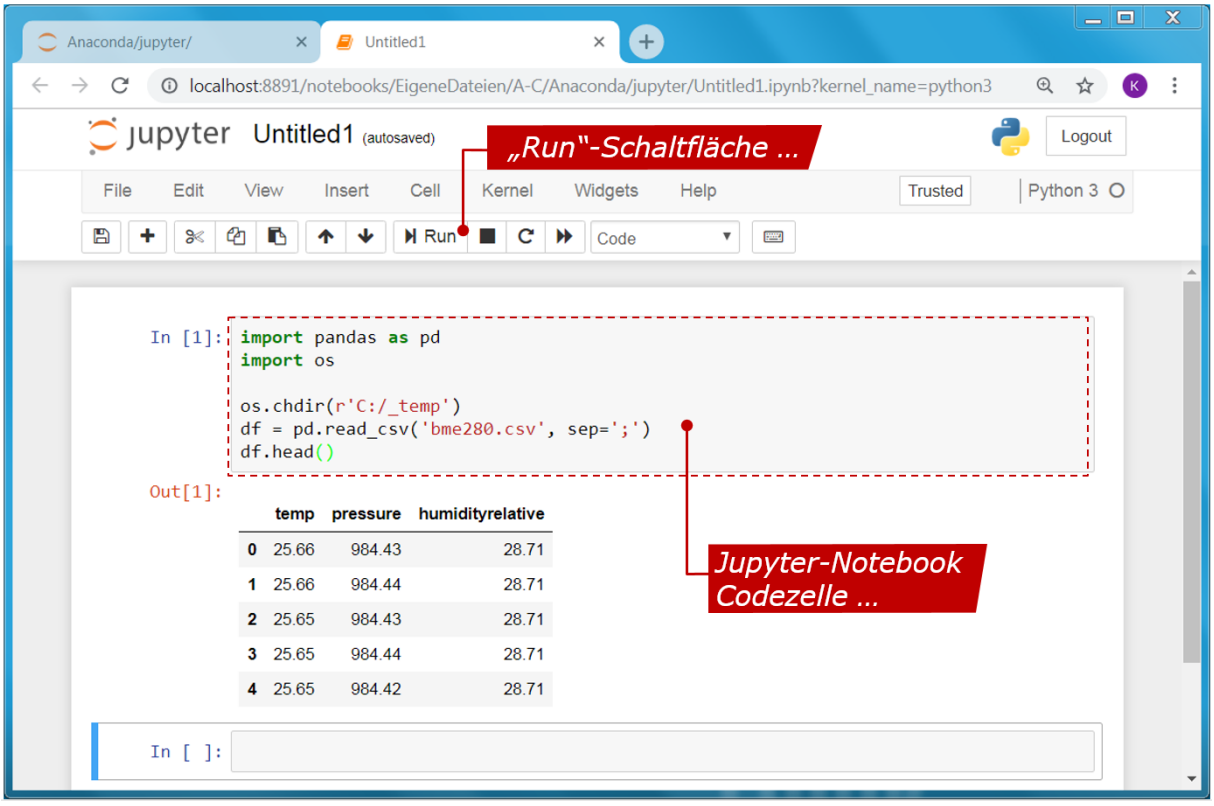
Betätigen Sie dann die Schaltfläche *New*, um ein neues Projekt zu erstellen. Wählen Sie im danach folgenden Dialogfenster bitte *Python3* aus.

Kopien Sie anschließend den Code aus dem Listing 7 in die erste Codezelle des Jupyter-Notebook-Eingabefensters. Betätigen Sie dann die *Run*-Schaltfläche, um die Ausführung des Codes im der Codezelle zu starten.

```
import pandas as pd
import os

os.chdir(r'C:/_temp')
df = pd.read_csv('bme280.csv', sep=';')
df.head()
```

Listing 7: Python-Quellcode für Jupyter Notebook, um die CSV-Datei aus „Die neunte Übung: CSV-Datei mit BME280-Messdaten erzeugen“ auf dem PC einzulesen und die ersten Zeilen anzuzeigen

Achtung: Wir gehen in diesem Text davon aus, dass sich die CSV-Datei *bme280.csv* im PC-Verzeichnis *C:/_temp* befindet. Passen Sie die Pfadangabe *C:/_temp* in den Python-Codes bitte an Ihre individuellen Bedürfnisse an, damit diese Beispiele auf Ihrem PC funktioniert.



In dem Beispiel zu Listing 7 wird die gesamte CSV-Datei mit Hilfe der Python-Pandas-Bibliothek in eine interne Tabellenstruktur eingelesen. Dort kann anschließend mittels zahlreicher Funktionen und Methoden auf die einzelnen Daten zugegriffen werden. Weitere Informationen zu *Pandas* finden Sie über den Link https://pandas.pydata.org/ im Internet. Verantwortlich für den gesamten Einlesevorgang ist die hier folgende Codezeile:

```
df = pd.read_csv('bme280.csv', sep=';')
```

Danach existiert ein sogenannter *Pandas Dataframe* mit dem Namen *df*. Ein solcher Dataframe ist eine zweidimensionale Datenstruktur mit Zeilen (Rows) und Spalten (Columns). Die einzelnen BME280-Messwerte zu Temperatur, Luftdruck und rel. Luftfeuchte bilden die Spalten mit den Namen *temp*, *pressure* und *humidityrelative*. Eine einzelne Zeile besteht aus je einem BME280-Messwert zu Temperatur, Luftdruck und rel. Luftfeuchte. Die Messwerte innerhalb der Zeile sind durch ein ‚;' voneinander getrennt.

Die einzelnen Zeilen eines Pandas-Dataframe besitzen jeweils eine fortlaufende Nummerierung als Index. Die erste Zeile hat unter Python standardmäßig den Index 0. Pandas bietet vielfältige Möglichkeiten, um mit Hilfe des Index in einem Dataframe zu navigieren und ausgewählte Datenelemente zu betrachten.

Eingabe	**Funktion**
`df`	Alle Zeilen und Spalten des Dataframe *df* ausgeben.
`df[0:10]`	Nur die Zeilen mit dem Index 0-9 des Dataframe *df* ausgeben.
`df[::2]`	Nur jede zweite Zeile des Dataframe *df* ausgeben.
`df.head()`	Die ersten Zeilen des Dataframe *df* ausgeben.
`df.head(10)`	Die ersten 10 Zeilen des Dataframe *df* ausgeben.
`df.tail()`	Die letzten Zeilen des Dataframe *df* ausgeben.
`df.tail(10)`	Die letzten 10 Zeilen des Dataframe *df* ausgeben.
`df.shape`	Anzahl der Zeilen und Spalten für den Dataframe *df* ausgeben.
`len(df)`	Anzahl der Zeilen für den Dataframe *df* ausgeben.
`len(df.columns)`	Anzahl der Spalten für den Dataframe *df* ausgeben.
`df['pressure'][7]`	Ausgabe des Datenwerts in der Spalte ‚pressure' für die Zeile 7.
`df['pressure'][0:10]`	Ausgabe der Datenwerte in der Spalte ‚pressure' für die Zeilen 0-9.
`df[0:10]['pressure']`	Ausgabe der Datenwerte in der Spalte ‚pressure' für die Zeilen 0-9.
`df.loc[0:3,'temp']`	Ausgabe der Datenwerte in der Spalte ‚temp' für die Zeilen 0-3.
`df.iloc[0:3,0]`	Ausgabe der Datenwerte in der Spalte ‚temp' für die Zeilen 0-3.
`df.iloc[:,0]`	Ausgabe aller Datenwerte in der Spalte ‚temp' des Dataframe *df*.

Tabelle 4: Beispiele für Funktionen und Methoden zur Anzeige eines Pandas-Dataframe

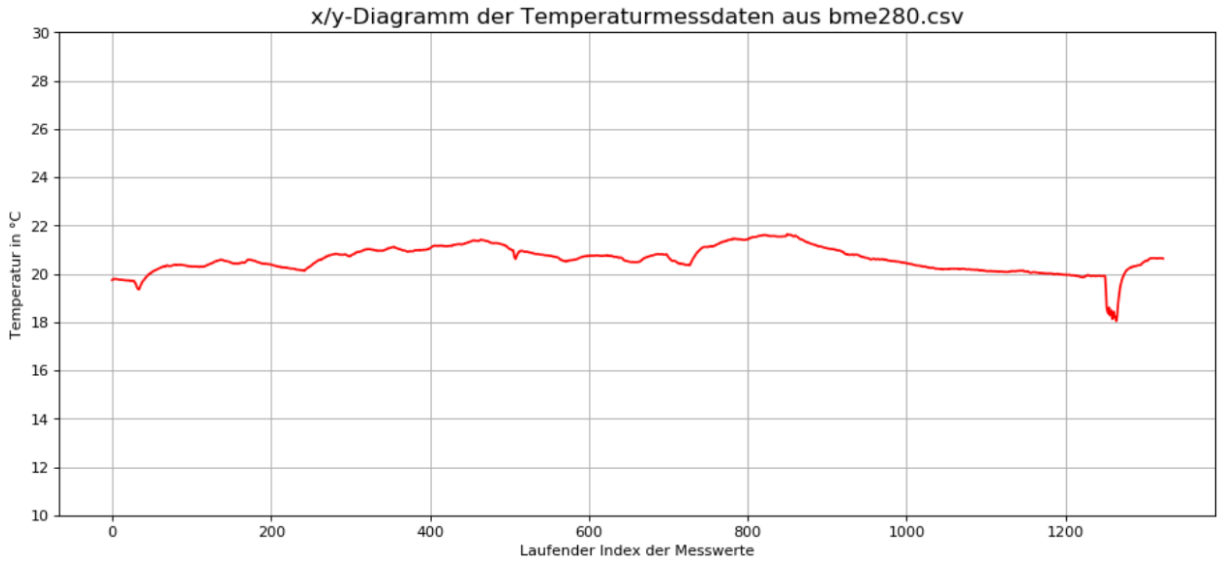
Wenn wir neben Pandas auch noch die Python-Matplotlib-Bibliothek nutzen, lassen sich beispielsweise die Datenelemente in einem Pandas-Dataframe als Liniendiagramm darstellen. Weitere Informationen zu *Matplotlib* findet man unter https://matplotlib.org/.

```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import os

os.chdir(r'C:/_temp')
df = pd.read_csv('bme280.csv', sep=';')
plt.figure(figsize=(14,6), dpi=80)
ax= plt.axes()
ax.set_ylim([10.0, 30.0])
ax.set_yticks([10.0, 12.0, 14.0, 16.0, 18.0, 20.0, 22.0, 24.0, 26.0,
28.0, 30.0])
plt.title("x/y-Diagramm der Temperaturmessdaten aus bme280.csv")
plt.xlabel('Laufender Index der Messwerte')
```

```
plt.ylabel('Temperatur in \xb0C')
df["temp"].plot(color= "red")
plt.grid()
plt.show()
```

Listing 8: Python-Quellcode für Jupyter Notebook, um die Temperatur aus der CSV-Datei *bme280.csv* in einem Liniendiagramm darzustellen

Kopien Sie den Python-Code aus dem Listing 8 in eine weitere Codezelle des Jupyter-Notebook-Eingabefensters. Betätigen Sie dann die *Run*-Schaltfläche, um die Ausführung in der neuen Codezelle zu starten. Danach müssten Sie ein Diagramm wie in der folgenden Abbildung als Jupyter-Note-book-Ausgabe erhalten:



Beachten Sie bitte, dass für die Abbildung eine CSV-Datei *bme280.csv* mit insgesamt über 1300 Messwerten verwendet wurde. Kleinere CSV-Dateien mit ca. 30 Messwerten ergeben in der Regel sehr einfache Diagramme mit wenig Aussagekraft.

Pandas unterstützt des Weiteren auch das Ermitteln typischer Kennzahlen für die deskriptive Statistik. Mit einer einzigen Funktion lassen sich beispielsweise Min, Max, Mittelwert, Standardabweichung usw. für einen Dataframe ermitteln:

```
import pandas as pd
import numpy as np
import os

os.chdir(r'C:/_temp')
df = pd.read_csv('bme280.csv', sep=';')
df.describe()
```

Listing 9: Durch die Codezeile `df.describe()` werden für den Dataframe *df* verschiedene Kennzahlen, wie zum Beispiel Minimum, Maximum, Mittelwert, Median usw. ermittelt und ausgegeben.

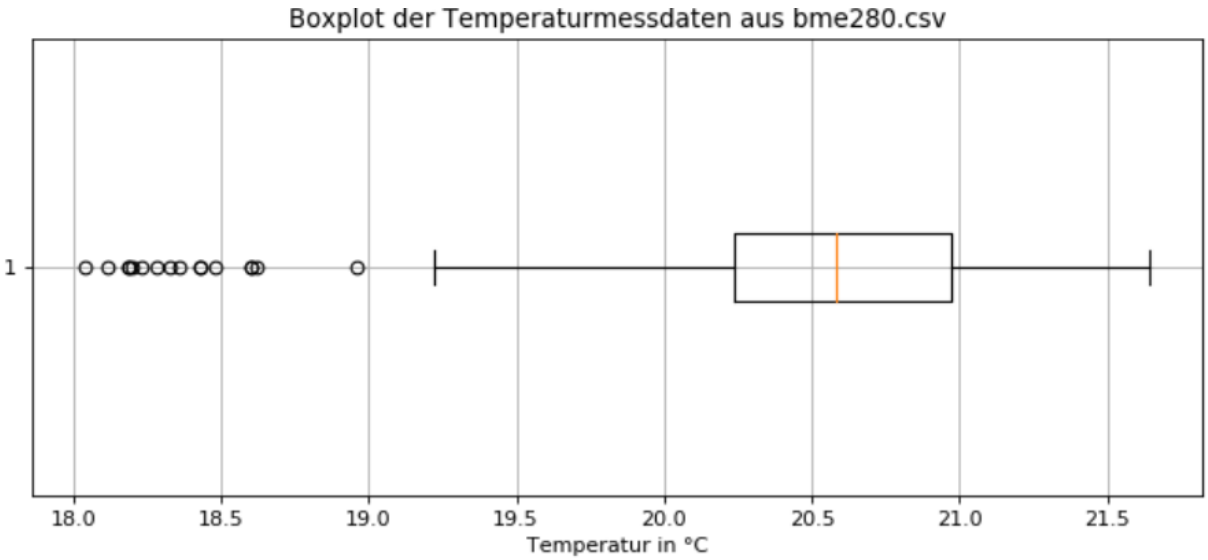
Neben den Kennzahlen der deskriptiven Statistik per Pandas lässt sich mit Hilfe der Matplotlib-Bibliothek auch ein Boxplot zu den Messwerten in der CSV-Datei *bme280.csv* erstellen. Das hier folgende Listing 10 liefet ein Beispiel für die BME280-Temperaturmesswerte des Pandas-Dataframe:

```
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import os

os.chdir(r'C:/_temp')
df = pd.read_csv('bme280.csv', sep=';')

plt.figure(figsize=(10,4), dpi=80)
ax= plt.axes()
plt.boxplot(df['temp'], vert=False)
plt.title("Boxplot der Temperaturmessdaten aus bme280.csv")
plt.xlabel('Temperatur in \xb0C')
plt.grid()
plt.show()
```

Listing 10: Python-Quellcode für Jupyter Notebook, um die Streuungs- und Lagemaße der Temperatur aus der CSV-Datei *bme280.csv* in einem Boxplot darzustellen

Kopien Sie den Python-Code aus dem Listing 10 in eine Jupyter-Notebook Codezelle und betätigen Sie dann die *Run*-Schaltfläche. Danach müssten Sie einen Boxplot wie in der folgenden Abbildung erhalten.

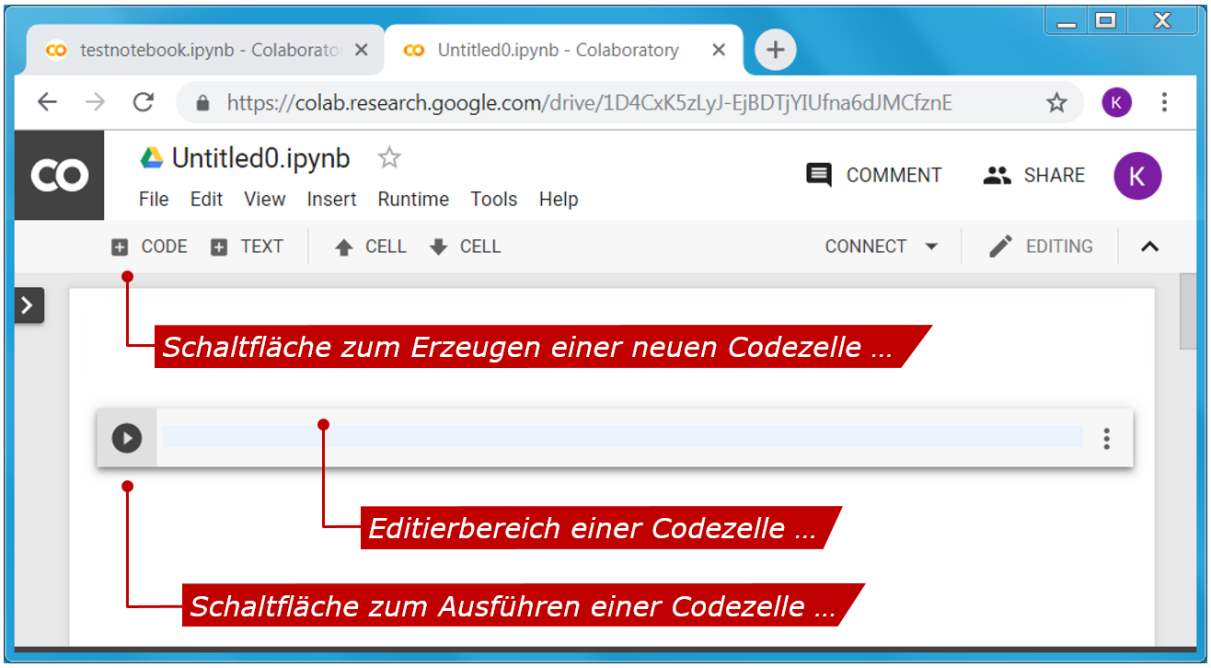
Die Jupyter-Notebook-Boxplot-Ausgabe liefert eine virtuelle Übersicht zu den Streuungs- und Lagemaßen der Temperaturmessungen aus der jeweiligen BME280-CSV-Datei (Beachten Sie bitte, dass für den folgenden Boxplot – wie bei dem x/y-Diagramm weiter oben – wieder eine BME280-Messreihe mit über 1300 Messwerten verwendet wurde).



Ein Boxplot fasst die Messwerte eines Datensatzes in fünf Punkten zusammen: Die Box selbst ist durch das untere und obere Quartil begrenzt. Sie umfasst den Bereich, in dem 50% aller Messwerte liegen. Innerhalb der Box ist der Median (50%-Wert) als Markierung zu finden. Er unterteilt die Box in zwei Hälften. An den beiden Enden der Box sind Antennen (Whisker) angefügt. Sie begrenzen einen Bereich, in dem z. B. 95% der Messwerte liegen (siehe hierzu auch https://de.wikipedia.org/wiki/Box-Plot). Außerhalb des oberen und unteren Whiskers existieren in einigen Fällen Ausreißer, die dann z. B. 5% der Messwerte ausmachen.

Die elfte Übung: CSV-Datei mit Cloud-Entwicklungsumgebung auswerten

Wenn Sie, aus welchen Gründen auch immer, die zuvor in der zehnten Übung angesprochene Open-Source-Data-Science-Entwicklungsumgebung *Anaconda* nicht auf Ihrem PC installieren wollen oder dürfen, können Sie mit Hilfe eines Webbrowsers auch direkt in der Cloud arbeiten. Eine sehr einfache und kostenlose Möglichkeit ist *Google CoLab*, siehe https://colab.research.google.com/. Sie benötigen dafür lediglich ein Google-Konto, was man bei Bedarf innerhalb weniger Minuten einrichten kann.



Wir werden daher in dieser Übung, analog zu „Die zehnte Übung: CSV-Datei mit PC-Entwicklungsumgebung auswerten“, die CSV-Datei *bme280.csv* in einen Pandas Dataframe einlesen und auswerten. Dafür nutzen wir in der Cloud Google CoLabs als Entwicklungsumgebung.

Zuerst muss die CSV-Datei *bme280.csv* von Ihrem PC aus in die Google-Cloud übertragen werden. Kopieren Sie dazu den Code aus dem Listing 11 in eine leere CoLab-Codezelle und führen diese Codezelle dann durch das Betätigen der dafür vorgesehenen Schaltfläche aus. Bei der Ausführung erhalten Sie einen Dateiauswahldialog, in dem Sie den Speicherort von *bme280.csv* auf Ihrem PC eintragen bzw. auswählen müssen. Danach erfolgt der automatische Upload dieser Datei in die Cloud.

```
from google.colab import files
uploaded = files.upload()
```

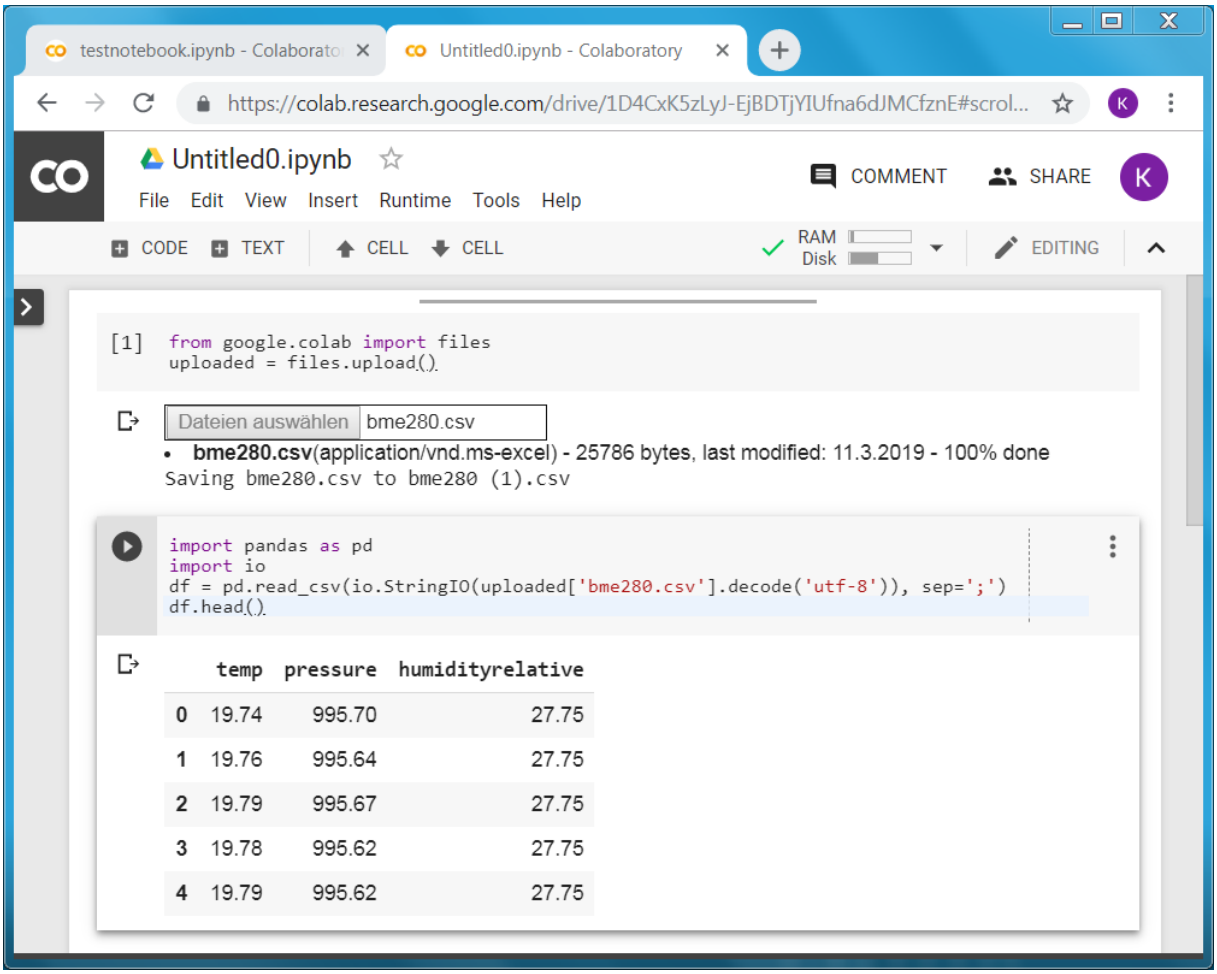
Listing 11: Python-Quellcode für Google CoLab, um die CSV-Datei aus „Die neunte Übung: CSV-Datei mit BME280-Messdaten erzeugen“ in die Cloud zu übertragen.

Nach dem erfolgreichen Upload von bme280.csv erzeugen Sie bitte eine neue, leere Codezelle in der CoLab-Arbeitsoberfläche und kopieren den Code aus Listing 12 in diese neue Zelle. Bringen Sie die Codezelle dann zur Ausführung. Beobachten Sie bitte die Ausgaben.

```
import pandas as pd
import io
df = pd.read_csv(io.StringIO(uploaded['bme280.csv'].decode('utf-
8')), sep=';')
```

```
df.head()
```

Listing 12: Die CSV-Datei wird zunächst in einen Pandas Dataframe eingelesen. Dann werden die ersten fünf Zeilen ausgegeben. Beachten Sie bitte, dass der `pd.read_csv()` Funktionsaufruf in Google CoLab andere Parameter benötigt, als z. B. unter Anaconda oder der DNP/AISS1-Python-Implementierung.

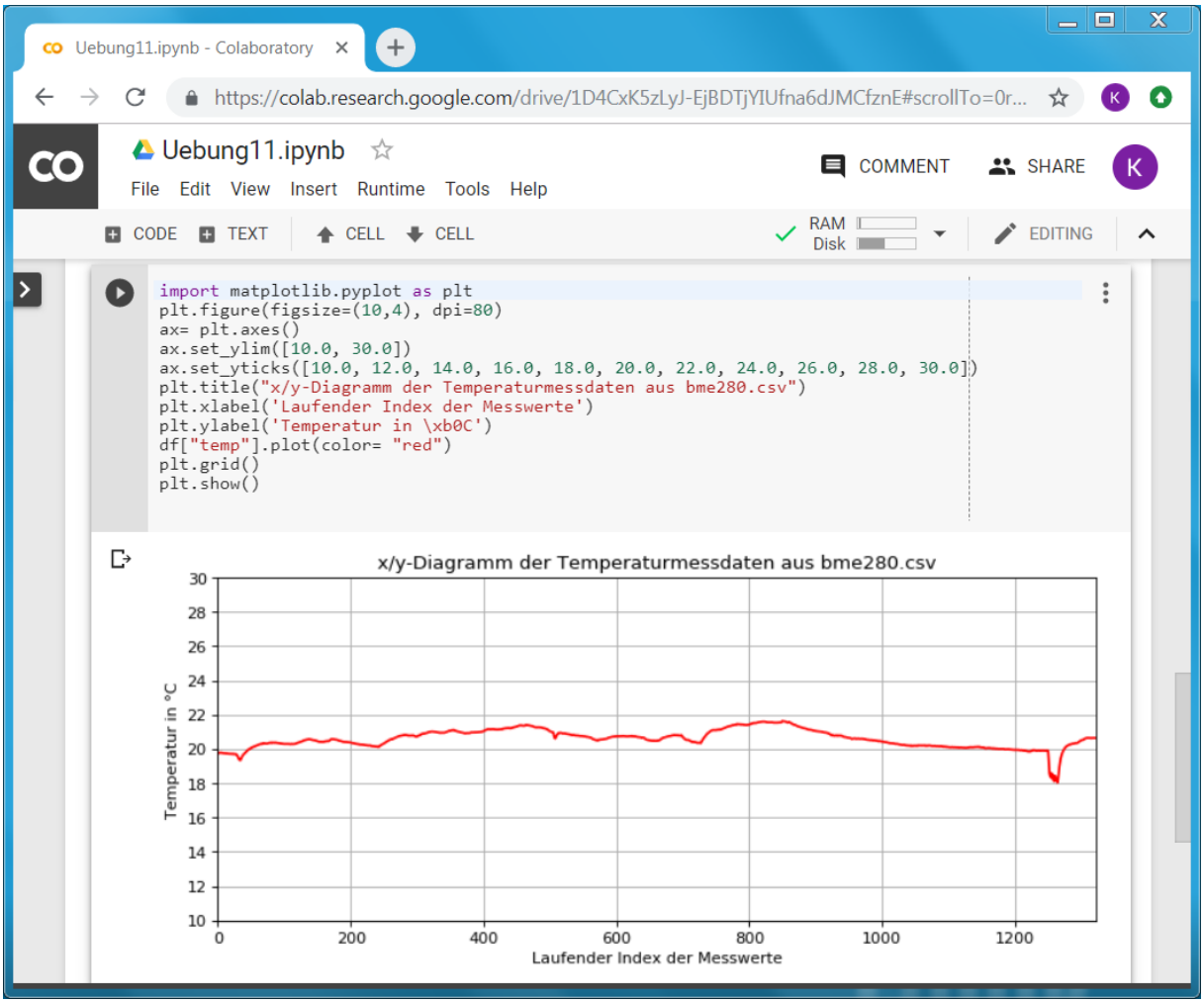


Neben Pandas bietet CoLab auch die Python-Matplotlib-Bibliothek, um die Daten der CSV-Datei in einem Diagramm zu visualisieren. Listing 13 liefert hierfür einen Beispiel-Code.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,4), dpi=80)
ax= plt.axes()
ax.set_ylim([10.0, 30.0])
ax.set_yticks([10.0, 12.0, 14.0, 16.0, 18.0, 20.0, 22.0, 24.0, 26.0,
28.0, 30.0])
plt.title("x/y-Diagramm der Temperaturmessdaten aus bme280.csv")
plt.xlabel('Laufender Index der Messwerte')
plt.ylabel('Temperatur in \xb0C')
df["temp"].plot(color= "red")
plt.grid()
plt.show()
```

Listing 13: Python-Quellcode für CoLab, um die Temperatur aus der CSV-Datei *bme280.csv* in einem Liniendiagramm darzustellen

Erzeugen Sie bitte eine neue Codezelle in der CoLab-Arbeitsoberfläche. Kopieren Sie dann den Code aus Listing 11 in diese Zelle und betätigen Sie danach die Schaltfläche zum Ausführen der Zelle.

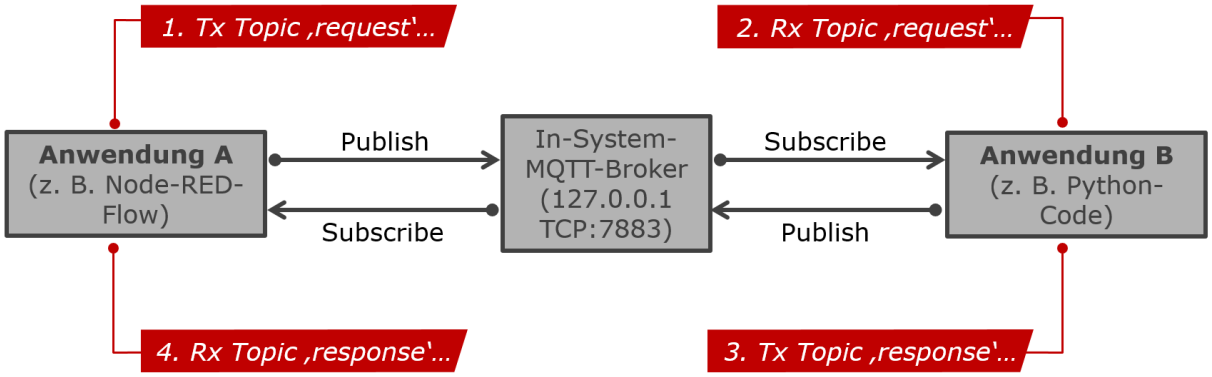


Google selbst bezeichnet CoLab als Umgebung für ausführbare Dokumente (Executable Documents), um Python-Code zu schreiben, auszuführen und mit anderen Entwicklern zu teilen. CoLab bietet eine nahezu identische Python3-Laufzeitumgebung und die gleichen Data-Science-Bibliotheken wie ein DNP/AISS1. Es ist daher eine einfach zu nutzende Entwicklungsumgebung, um Sensordaten ein Projekt zu untersuchen, ein Machine-Learning-Modell zu erzeugen und zu validieren. Der dabei entstehende Python-Code lässt sich danach auf dem DNP/AISS1 nutzen.

Die zwölfte Übung: In-System-MQTT-Broker (ISMB) nutzen

In der Praxis gibt es Machine-Learning-Anwendungen, die zum Beispiel sowohl Python-Code als auch Node-RED-Flows nutzen und die miteinander kommunizieren müssen. Ein typischer Anwendungsfall wäre die Weitergabe aktueller Sensordaten an einem Algorithmus zur Datenanalyse.

Um eine solche Kommunikation zu ermöglichen, ist auf dem DNP/AISS1 ein *In-System-MQTT-Broker (ISMB)* vorinstalliert, der über die IP-Adresse **127.0.0.1** und den TCP-Port **7883** zu erreichen ist.

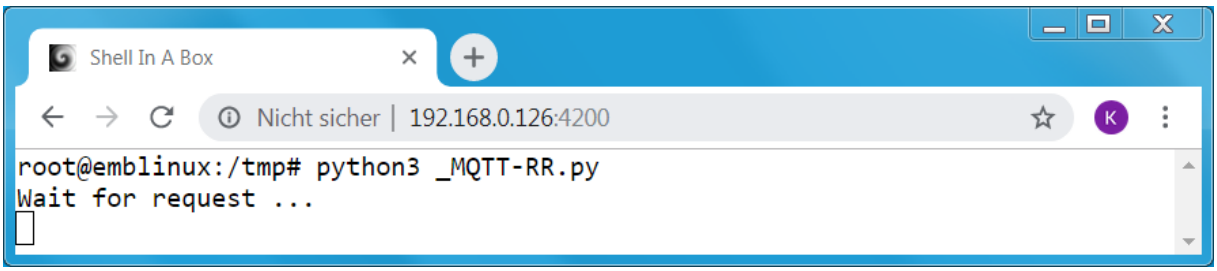


Um sich mit den ISMB-Funktionen vertraut zu machen, stehen zwei Dateien und ein spezielles Hilfsverzeichnis als MQTT-basiertes Request-Response-Beispiel zur Verfügung.

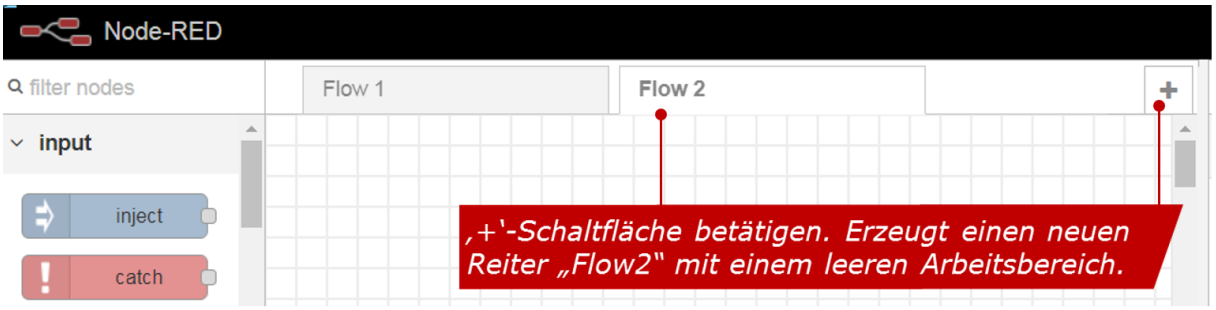
Name	Typ	Funktion
`_MQTT-RR.json`	Datei	Node-RED-Flow mit vier Nodes, um einen Request per Publish an den ISMB zu schicken. Dafür wird der MQTT-Topic *request* benutzt. Gleichzeitig wird über den Topic *response* auf eine Nachricht gewartet, die über einen Debug Node im dazugehörenden Ausgabebereich dargestellt wird.
`_MQTT-RR.py`	Datei	Python-Anwendung. Es wird in einer Endlosschleife per MQTT Subscribe zunächst auf einen Request über den Topic *request* gewartet. Anschließend wird eine OK-Antwort per MQTT Publish über den Topic *response* versendet.
`paho`	Verzeichnis	Verzeichnis mit den MQTT-Funktionen für Python-Programme (wird z. B. als Bibliothek in `_MQTT-RR.py` eingebunden).

Tabelle 5: Die einzelnen Bestandteile des Request-Response-Beispiels

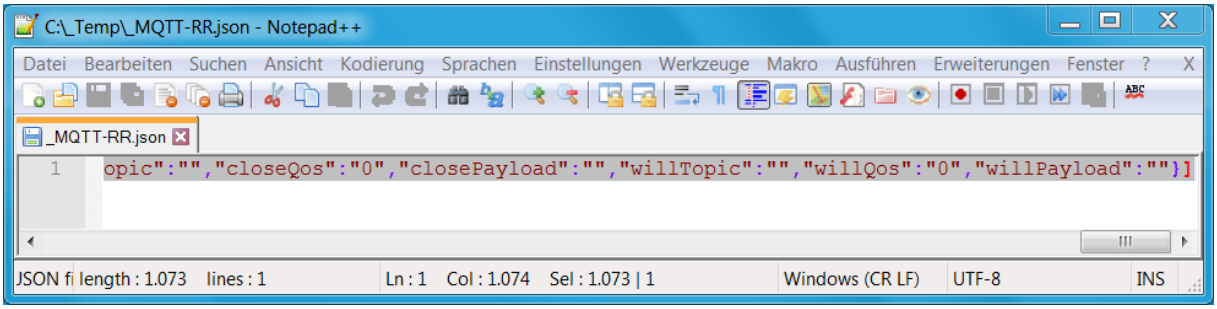
Laden Sie zunächst die Datei *_MQTT-RR.py* und das Verzeichnis *paho* mit Hilfe des FTP-Clients vom PC aus in das DNP/AISS1-Verzeichnis /tmp. Bringen Sie die Quellcodedatei *_MQTT-RR.py* dann zur Ausführung. Benutzen Sie „Die fünfte Übung: Erste Schritte mit Python“ als Vorlage für Ihre Vorgehensweise.



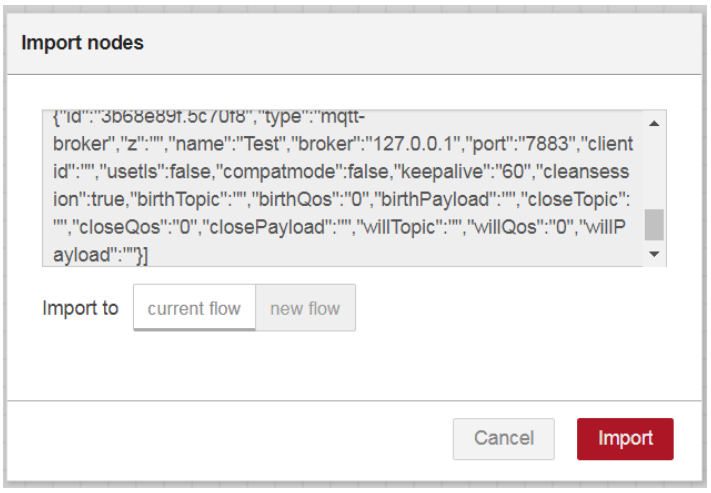
Greifen Sie per Webbrowser auf die Node-RED-Benutzeroberfläche des DNP/AISS1 zu. Klicken Sie auf die ‚+'-Schaltfläche, um einen neuen Reiter mit einer leeren Arbeitsoberfläche zu erzeugen. Der neue Arbeitsbereich trägt den Namen *Flow2*.



Öffnen Sie nun auf Ihrem PC mit einem Texteditor zum Bearbeiten von Quellcodes die Datei *_MQTT-RR.json*. Markieren Sie den gesamten Text und übernehmen Sie den Text in das Clipboard Ihres PCs, z. B. per CTRL-C. **Achtung:** Die gesamte Datei *_MQTT-RR.json* besteht nur aus einer einzigen Textzeile mit über 1.000 Zeihen.

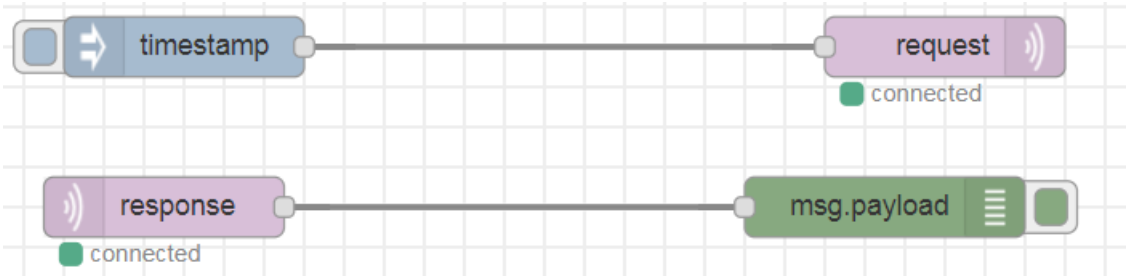


Öffnen Sie dann in Ihrem Webbrowser das Node-RED-Menü in der Node-RED-Benutzeroberfläche und wählen Sie dort die *Import*-Funktion aus. Wählen Sie im dann sichtbaren Untermenü den Punkt *Clipboard* aus.



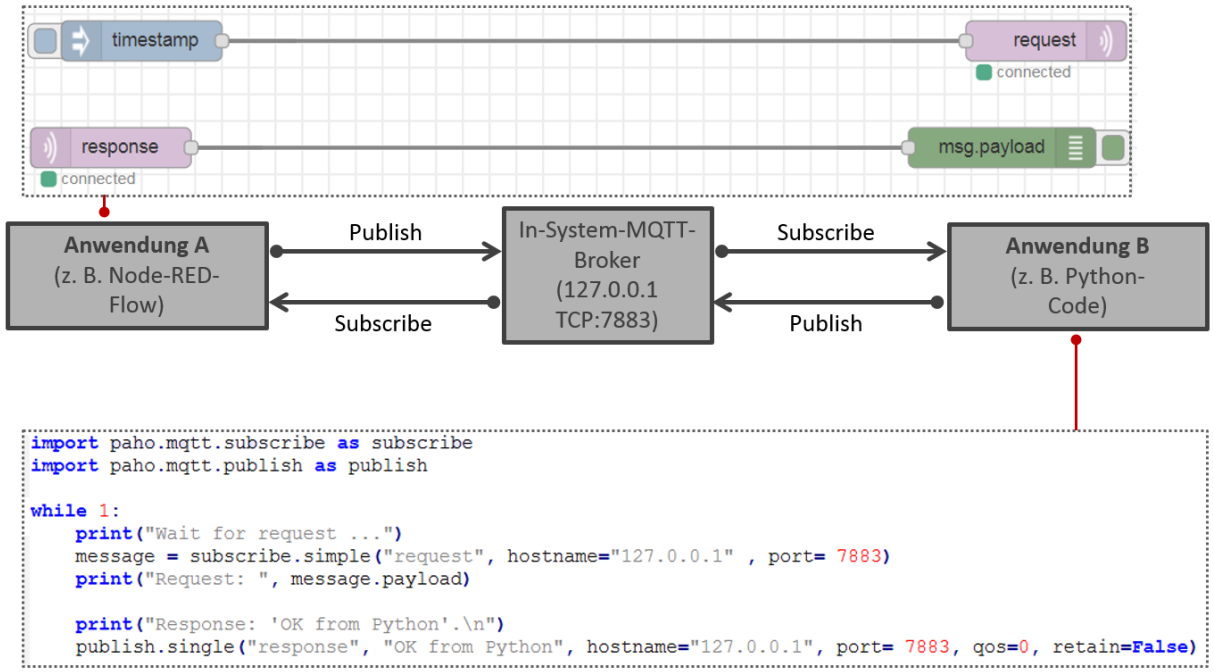
Übertragen Sie per CTRL-V den zuvor vom Editor aus in das PC-Clipboard kopierten JSON-Text in das nun sichtbare *Import nodes*-Fenster (siehe Abbildung) und betätigen Sie dann die *Import*-Schaltfläche.

Sie haben nun einen kompletten Node-RED-Flow mit vier Nodes aus einer Datei in den Arbeitsbereich *Flow2* kopiert. Positionieren Sie den Flow entsprechend Ihren Wünschen und starten Sie den neuen Flow danach über die *Deploy*-Schaltfläche.



Oben links ist ein *Inject Node* platziert, der beim Aktivieren (Triggern über die dafür vorgesehene Schaltfläche) einen Zeitstempel mit der aktuellen Uhrzeit an den *MQTT Output Node* oben rechts überträgt. Der MQTT Output Node ist für den In-System-MQTT-Broker des DNP/AISS1 (IP-Adresse 127.0.0.1, TCP-Port 7883) und den Topic *request* vorkonfiguriert. Die Funktion eines MQTT Output Node entspricht einer MQTT-Publish-Operation (in diesem Fall ein Publish an den Topic *request*).

Unten links in dem Node-RED-Flow ist ein *MQTT Input Node* zu finden. Dieser Node ist ebenfalls für den In-System-MQTT-Broker vorkonfiguriert, allerdings wird hier der Topic *response* verwendet. Erhält der MQTT Input Node neue Daten von ISMB, werden diese an den *Debug Node* unten rechts weitergeleitet. Die Funktion eines MQTT Input Node entspricht einer MQTT-Subscribe-Operation (hier ein Subscribe auf den Topic *response*).

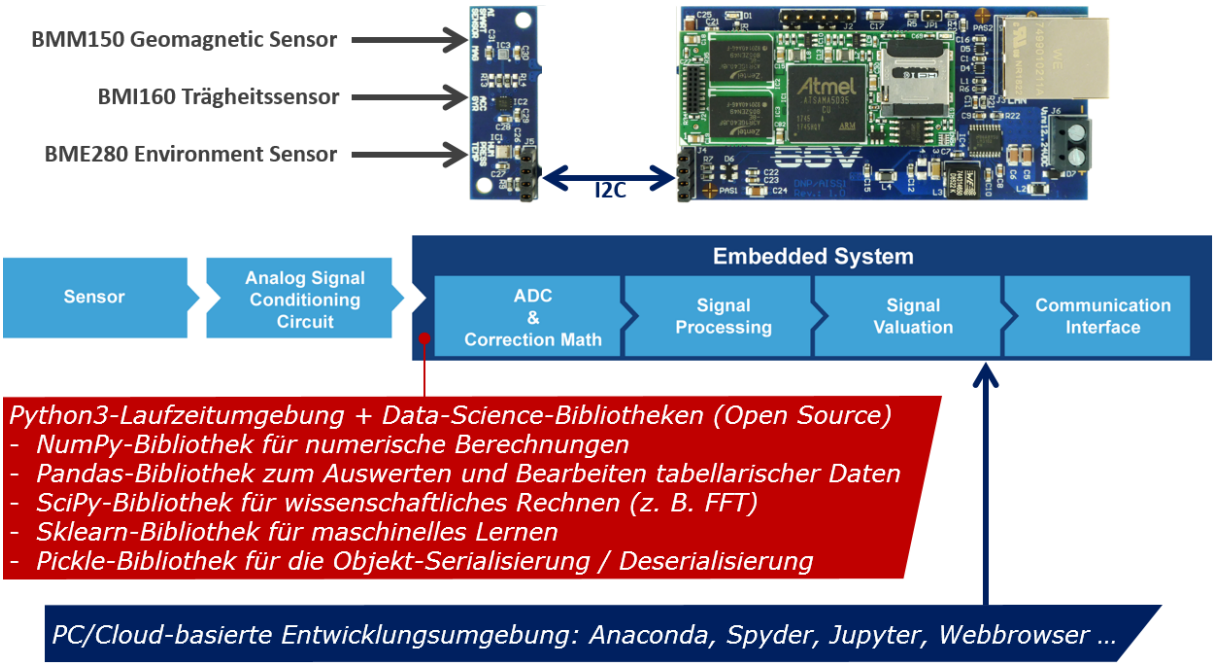



```
import paho.mqtt.subscribe as subscribe
import paho.mqtt.publish as publish

while 1:
    print("Wait for request ...")
    message = subscribe.simple("request", hostname="127.0.0.1" , port= 7883)
    print("Request: ", message.payload)

    print("Response: 'OK from Python'.\n")
    publish.single("response", "OK from Python", hostname="127.0.0.1", port= 7883, qos=0, retain=False)
```

Klicken Sie nun die *Trigger-Schaltfläche* des Inject Nodes und beobachten sie dabei die jeweiligen Ausgaben des Debug Nodes im dazugehörenden Ausgabebereich der Node-RED-Benutzeroberfläche.

Die dreizehnte Übung: Data-Science-Bibliotheken nutzen

Ein DNP/AISS1 wird mit vorinstallierter Python3-Laufzeitumgebung und verschiedenen Data-Science-Bibliotheken ausgeliefert. Die meisten dieser Bibliotheken stehen auch in Python3-Entwicklungsumgebungen zu Verfügung. Dadurch lässt sich der Python-Code für eine Machine-Learning-Anwendung vor dem Einsatz auf einem DNP/AISS1 in einer Entwicklungsumgebung nahezu vollständig testen.



In der Tabelle 6 finden Sie eine kurze Übersicht der drei wichtigsten Python-Data-Science-Bibliotheken für das maschinelle Lernen in DNP/AISS1-basierten Projekten. Diese Bibliotheken werden auf jeden Fall auch in dem jeweils zum Einsatz kommenden Entwicklungssystem benötigt. Darüber hinaus ist in der Entwicklungsumgebung auch noch die Matplotlib-Bibliothek zur Datenvisualisierung mittels geeigneter Diagramme erforderlich.

Bibliothek	Funktion
NumPy	Funktionen zur Erzeugung und Verarbeitung mehrdimensionaler Arrays. Neben den Datenstrukturen bietet NumPy auch sehr effiziente Funktionen für numerische Berechnungen. Weitere Informationen und eine umfangreiche User Guide findet man unter www.numpy.org.
Pandas	Funktionen zur Verwaltung und Datenanalyse für Tabellen und Zeitreihendaten. In einen Pandas Dataframe lassen sich beispielsweise komplette CSV-Dateien einlesen, analysieren und bearbeiten. Weitere Informationen und eine User Guide findet man unter https://pandas.pydata.org/.
Sklearn	Die Bibliothek Scikit-learn beinhaltet verschiedene Funktionen für das maschinelle Lernen. Insgesamt stehen verschiedenen Klassifizierungs- und Regressionsalgorithmen für Supervised Machine Learning sowie Clustering-Algorithmen zum Unsupervised Machine Learning zur Verfügung. Weitere Informationen und eine User Guide findet man unter https://scikit-learn.org/stable/

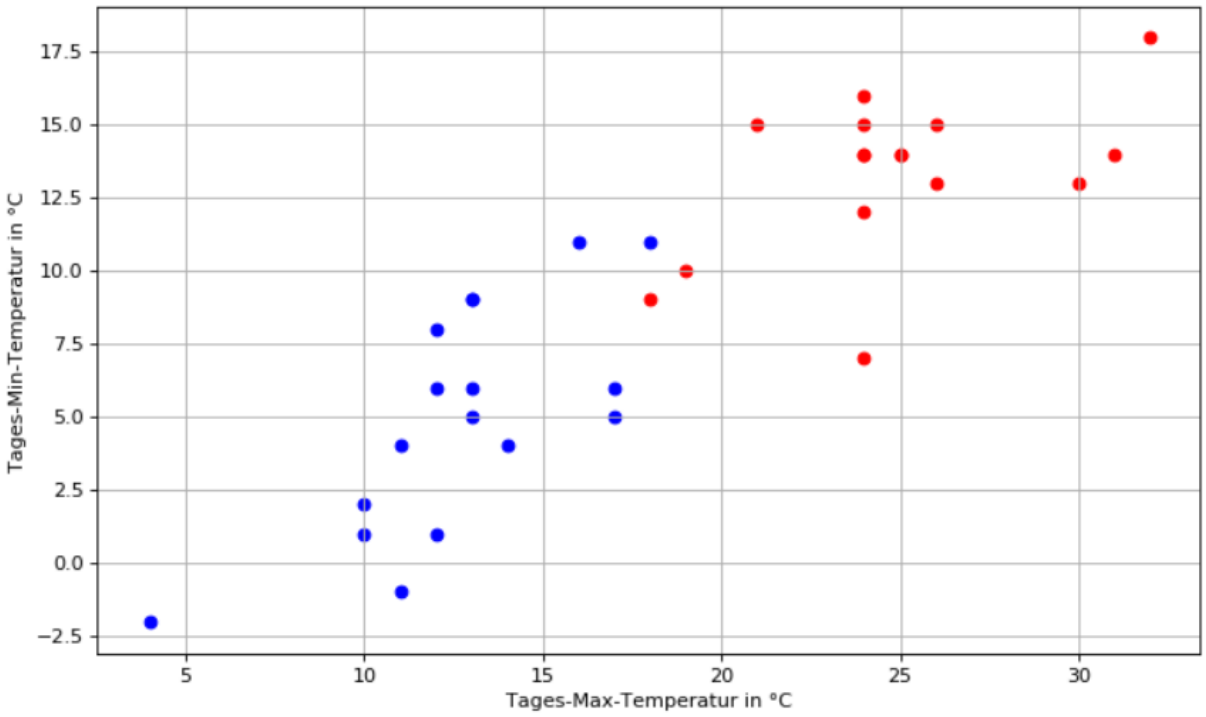
Tabelle 6: Ein DNP/AISS1 wird mit vorinstallierter Python3-Laufzeitumgebung und verschiedenen Data-Science-Bibliotheken ausgeliefert. Die drei wichtigen Bibliotheken für das maschinelle Lernen sind Numpy, Pandas und Sklearn (Scikit-learn).

Das hier folgende Listing 14 liefert ein Beispiel für die NumPy- und Matplotlib-Nutzung auf einem Entwicklungsrechner. Am Code-Anfang werden zunächst zwei NumPy-Arrays mit den Namen *summer* und *winter* erzeugt. Jedes Array-Element (z. B. `[21,15]`) bildet jeweils einen zweidimensionmallen Datenvektor mit der Höchst- (Max) und Tiefsttemperatur (Min) eines Sommer- oder Wintertages.

```
import numpy as np
summer= np.array([[24,16], [21,15], [24,14], [25,14], [25,14],
                  [26,13], [30,13], [31,14], [32,18], [24,14],
                  [24,12], [19,10], [18,9],  [24,7],  [24,15],
                  [26,15]])

winter= np.array([[11,4], [12,6], [11,-1], [12,1], [12,8], [17,5],
                  [17,6], [14,4], [13,6],  [13,9], [16,11],
                  [18,11],[13,9], [13,5],  [10,2], [10,1],
                  [4,-2]])

import matplotlib.pyplot as plt
plt.figure(figsize=(10,6), dpi=80)
plt.scatter(summer[:,0], summer[:,1], c= "r", alpha= 1.0)
plt.scatter(winter[:,0], winter[:,1], c= "b", alpha= 1.0)
plt.xlabel('Tages-Max-Temperatur in \xb0C')
plt.ylabel('Tages-Min-Temperatur in \xb0C')
plt.grid(True)
plt.show()
```

Listing 14: Beispiel für ein NumPy-Array mit zweidimensionalen Datenvektoren und die Darstellung als Scatterplot (Steudiagramm) per Matplotlib

Durch den weiteren Matplotlib-Code im Listing 14 werden die beiden Arrays *summer* und *winter* mit allen Datenvektorelementen in einem Scatterplot (Streudiagramm – siehe die hier folgende Abbildung) dargestellt. Jeder einzelne zweidimensionale Datenvektor bildet einen Datenpunkt in der Fläche des Streudiagramms. Die blaue Punktewolke entspricht den Wintertemperatur-, die rote den Sommertemperaturwerten.



Man kann in dem Streudiagramm relativ gut die beiden unterschiedlichen Positionsbereiche für die Vektoren aus den Arrays *summer* und *winter* erkennen. Jeder Bereich bildet eine deutlich sichtbare Klasse. Zur Trennung der Klassen könnte man eine senkrechte Linie bei x = 17,5 durch das Diagramm ziehen. Bis auf einen einzigen Fehler (blauer Datenpunkt bei [18,11]) würde diese Linie als optischer Klassifikator zwischen *summer* und *winter* dienen.

Listing 15 enthält den vollständigen Code für ein Beispiel, um per Supervised Machine Learning einen Klassifikator zu schaffen und mit den Datenvektorelementen aus den Arrays *summer* und *winter* zu trainieren. Durch dieses Training entsteht ein Modell, dass zur Klassifizierung bisher unbekannter Datenvektoren mit Max- und Min-Temperaturwerten dient. Als Klassifizierungsalgorithmus wird im Listing 15 ein Decision Tree (Entscheidungsbaum) verwendet. Dieser Algorithmus arbeitet mit sehr einfach nachvollziehbaren Entscheidungen, um unbekannte Daten gemäß dem zuvor aus den Trainingsdaten erlernten Mustern zu klassifizieren.

```
import numpy as np
X= np.array([[24,16], [21,15], [24,14], [25,14], [25,14], [26,13],
             [30,13], [31,14], [32,18], [24,14], [24,12], [19,10],
             [18,9],  [24,7],  [24,15], [26,15], [11,4],  [12,6],
             [11,-1], [12,1],  [12,8],  [17,5],  [17,6],  [14,4],
             [13,6],  [13,9],  [16,11], [18,11], [13,9],  [13,5],
             [10,2],  [10,1],  [4,-2]])

y= np.array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])

from sklearn import tree
clf= tree.DecisionTreeClassifier()
clf.fit(X, y)
```

Listing 15: Training und Modellbildung für einen Decision Tree-Klassifikator. Das Array X enthält die zusammengefassten Arrays *summer* und *winter* aus dem Listing 14. Über das Array y werden alle Datenvektorelemente von X jeweils einer Klasse (1 = Winter, 2 = Sommer) zugeordnet,

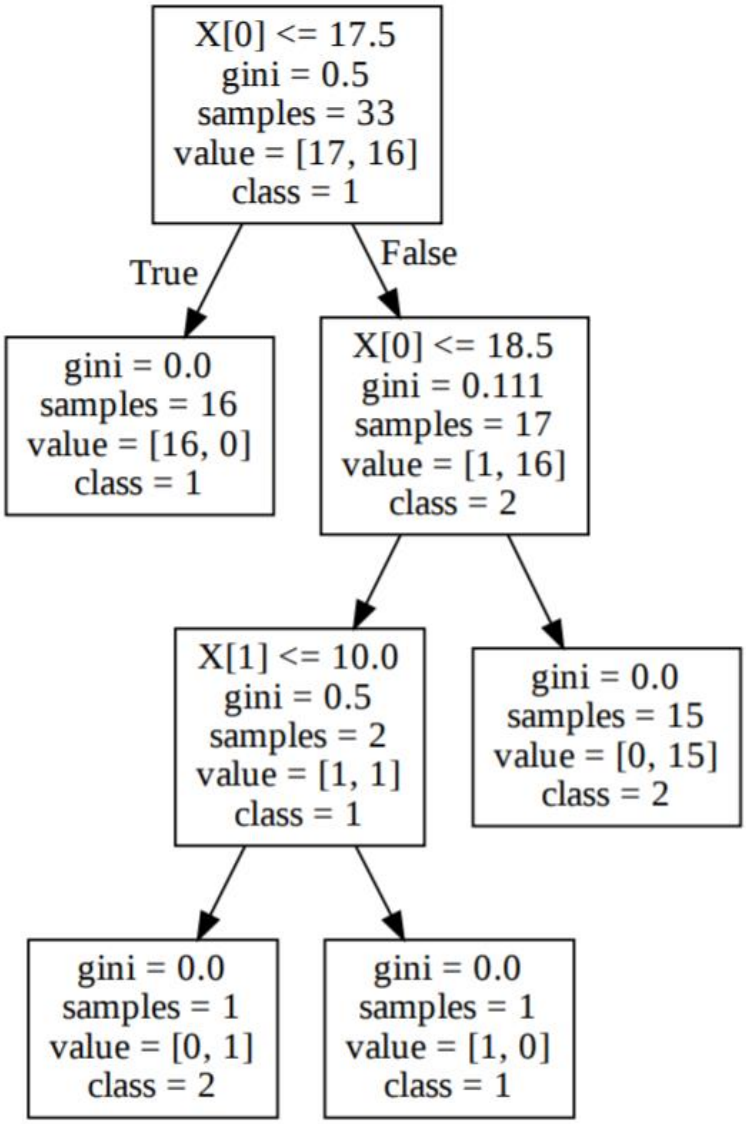
Die beiden Arrays *summer* und *winter* wurden im Listing 15 zu einem einzigen Array *X* zusammengefasst. Insgesamt enthält *X* genau 33 Datenvektorelemente. Um den Decision Tree-Klassifikator zu trainieren, muss allen Datenelementen in *X* jeweils ein Label zugeordnet werden, aus dem hervorgeht, zu welcher Klasse (*summer* und *winter*) ein Datenvektorelement von *X* gehört. Diese Zuordnung ermöglicht im Listing 15 das NumPy-Array *y*. Dieses Array dient somit zum Labeln der Lern- bzw. Trainingsdaten.

```
import graphviz
y_names= ["1", "2"]
dot_data= tree.export_graphviz(clf,
                               out_file=None,
                               class_names=y_names)
graph= graphviz.Source(dot_data)
graph.render("dt")
```

Listing 16: Das gesamte Klassifizierungsmodell des Listing 15 ist im Klassifikator `clf` gespeichert. Der dazu gehörende Entscheidungsbaum lässt sich in einer Grafik darstellen.

Kopieren sie zunächst den Code aus dem Listing 15 in eine Jupyter-Codezelle und führen Sie den Code dann aus. Erzeugen Sie danach eine neue Codezelle für den Beispielcode aus dem Listing 16

und führen Sie diese Zelle anschließend ebenfalls aus. Dadurch wird eine PDF-Ausgabe mit dem hier folgenden Entscheidungsbaum erzeugt.



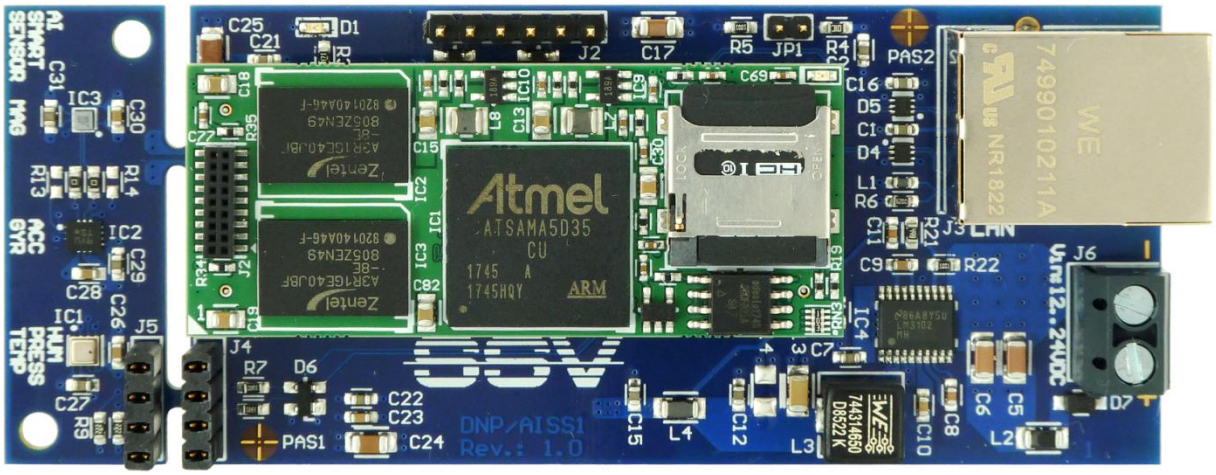
Beachten Sie bitte, dass die im Listing 16 benutzte Graphviz-Bibliothek nicht in allen Jupyter-Entwicklungsumgebungen vorinstalliert zur Verfügung steht. Diese Bibliothek muss bei Bedarf nachinstalliert werden.

```
clf.predict(np.array([[18.9,10.95]])
```

Listing 17: Codezeile, um mit Hilfe des zum Klassifikator `clf` gehörenden Entscheidungsbaums für das Wertepaar `[18.9,10.95]` eine Klasse vorherzusagen

Um neue Datenvektorelemente an Hand des Entscheidungsbaums einer Klasse (1 = Winter, 2 = Sommer) zuzuordnen, führen Sie bitte die Codezeile aus dem Listing 17 in einer weiteren neuen Codezelle aus. Verändern Sie das Wertepaar `[18.9,10.95]` und wiederholen Sie die Ausführung. Sie erhalten nach jeder Ausführung die durch den Entscheidungsbaum jeweils bestimme Klasse.

Anhang 1: Technische Daten DNP/AISS1

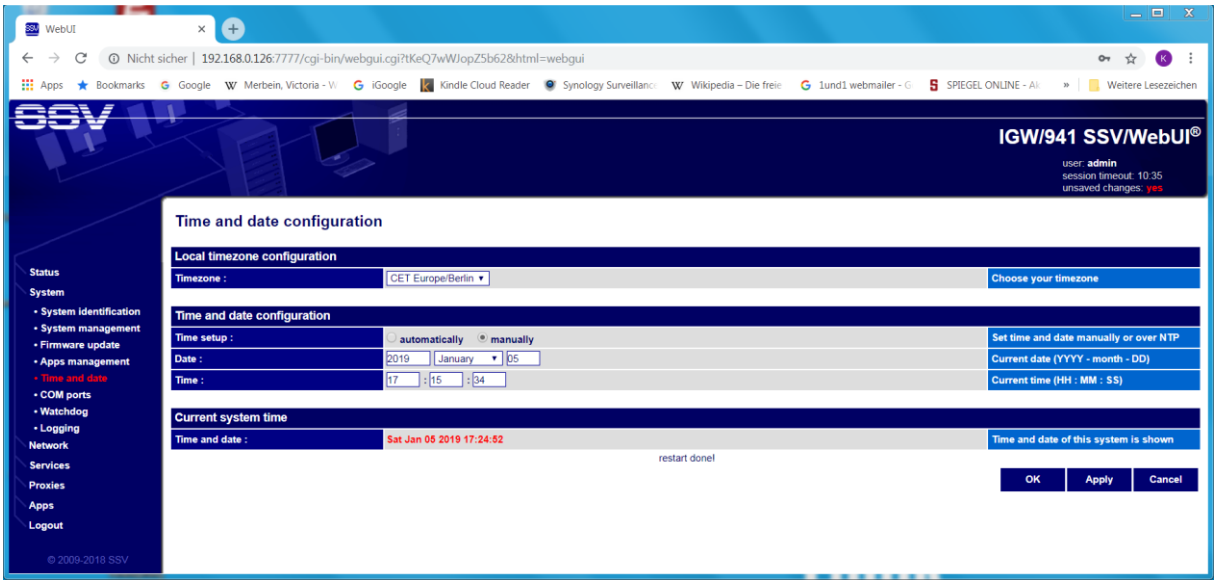


- 1x 10/100 Mbps Ethernet LAN
- 1x On-board I2C (trennbare) Verbindung zum Sensorboard
- 1x DIL/NetPC DNP/9535 Embedded Linux Microrechner:
  … ARM ATSAM-A5D35 MCU @ 528 MHz
  … 256 MB 32-bit DDR2 RAM @ 133 MHz
  … 4 MB Serial NOR Flash mit Boot Loader mit A/B-Bootkonzept für SD-Karte
  … 1x Micro-SD-Karte mit Linux-Betriebssystem und Firmware
- Gesamtabmessungen der Leiterplatte 102 x 40 mm
- Versorgungsspannung 12 - 24 VDC, max. 2 Watt
- Betriebstemperaturbereich 0 - 70 ˚C
- Lagertemperatur -40 - 85 ˚C
- Software auf der SD-Karte:
  … Embedded Linux Betriebssystem und Gateway Firmware
  … Web-basierte Konfigurationsoberfläche mit Passwortschutz
  … Einschaltbare Firmware-Updates von vorgegebener Serveradresse
  … I2C Treiber plus Analog Devices LibIIO
  … AISS Technology Stack mit Python-basierter KI-Software
  … SSV VPN (Virtual Private Network) für hochsichere Fernzugriffe
  … Node-RED, OPC UA u.v.a. als nachinstallierbare Apps (teilweise kostenpflichtig)
  … In-System-MQTT-Broker (ISMB) unter IP-Adresse 127.0.0.1, TCP-Port 7883
  … vorinstalliertes Machine-Learning-Beispiel mit Sensordaten, Node-RED-Integration und Erläuterung
- Sensorboard (trennbar) mit drei Sensoren und I2C-Verbindung zum DNP/9535:
  … 1x Bosch BMM150 Geomagnetic Sensor
  … 1x Bosch BMI160 6-Achsen Inertial Measurement Unit (Trägheitssensor)
  … 1x Bosch BME280 Environment Sensor (Temperatur, Luftdruck, rel. Luftfeuchte)
  … vollständige Integration der Sensoren in die Analog Devices LibIIO, Zugriff per I2C

Anhang 2: Uhrzeit manuell setzen

Der DNP/9535 Embedded Linux Microrechner des DNP/AISS1 besitzt eine Real Time Clock. Diese Uhr kann wahlweise über einen externen NTP-Server oder manuell mit den jeweils aktuellen Datum- und Zeitangaben synchronisiert werden.

Rufen Sie per Webbrowser über den Link http://192.168.0.126:7777 die Web-basierte Benutzeroberfläche des DNP/AISS1 auf und melden Sie sich mit einem gültigen Benutzernamen und dem dazu gehörenden Passwort an.



Wählen Sie dann im Menüpunkt *System* den Unterpunkt *Time and date* aus. Klicken Sie danach in der Zeile für den *Time setup* auf *manually*.

Betätigen Sie im Anschluss daran per Mausklick die *Apply*-Schaltfläche im unteren rechten Bereich des Browser-Fensters und warten Sie einen Moment. Danach wird das aktuelle Datum und die Uhrzeit automatisch in die Real Time Clock des DNP/9535 Embedded Linux Micro-rechners übernommen.

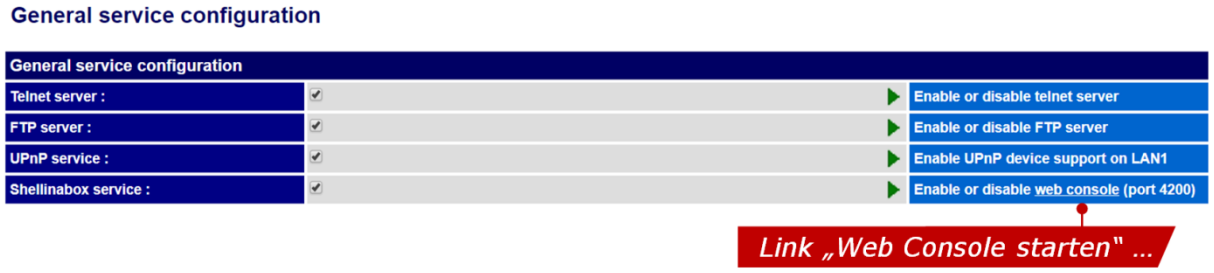
Beachten Sie bitte, dass die Real Time Clock des DNP/9535 keine Pufferbatterie besitzt. Durch das Trennen des DNP/AISS1 von der Versorgungsspannung gehen die voreingestellten Werte für Datum und Uhrzeit wieder verloren.

Anhang 3: Sensor-Integration in das DNP/9535-Linux

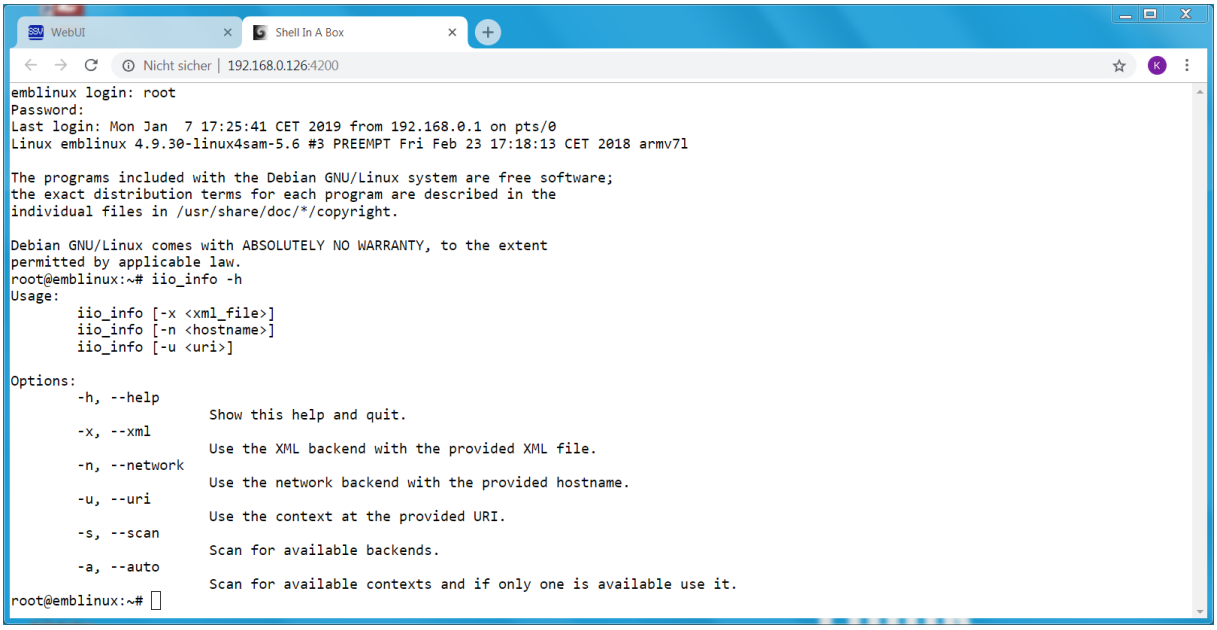
Die Bosch-Sensoren des DNP/AISS1 sind per Industrial I/O-Treiber in das Linux-Betriebssystem des DNP/9535 Microrechners eingebunden (siehe https://wiki.analog.com/software/linux/docs/iio/iio).

Um sich eine Übersicht zur internen Datenstruktur zu verschaffen, rufen Sie bitte per Webbrowser über den Link http://192.168.0.126:7777 die Web-basierte Benutzeroberfläche des DNP/AISS1 auf und melden Sie sich mit einem gültigen Benutzernamen und dem dazu gehörenden Passwort an.

Wählen Sie nun im Menüpunkt *Services* den Unterpunkt *General* aus. Klicken Sie danach auf den Link für die *Web Console*. Dadurch wird in einem weiteren Browser-Fenster der *Shell-In-a-Box Service* gestartet.



Melden Sie sich dann zunächst mit einem gültigen Benutzernamen und dem dazu gehörenden Passwort in der Web Console an.



Geben Sie nun das Kommando *iio-info -h* ein. Sie erhalten dadurch eine Übersicht über die einzelnen Funktionen des Hilfsprogramms *iio-info* zum Industrial I/O-Treiber. Geben Sie danach *iio-info –n 127.0.0.1* ein. Mit diesen Parametern wird eine längere Übersicht zu den erkannten Sensoren und den verfügbaren Datenkanälen (Channels) der jeweiligen Sensoren ausgegeben.