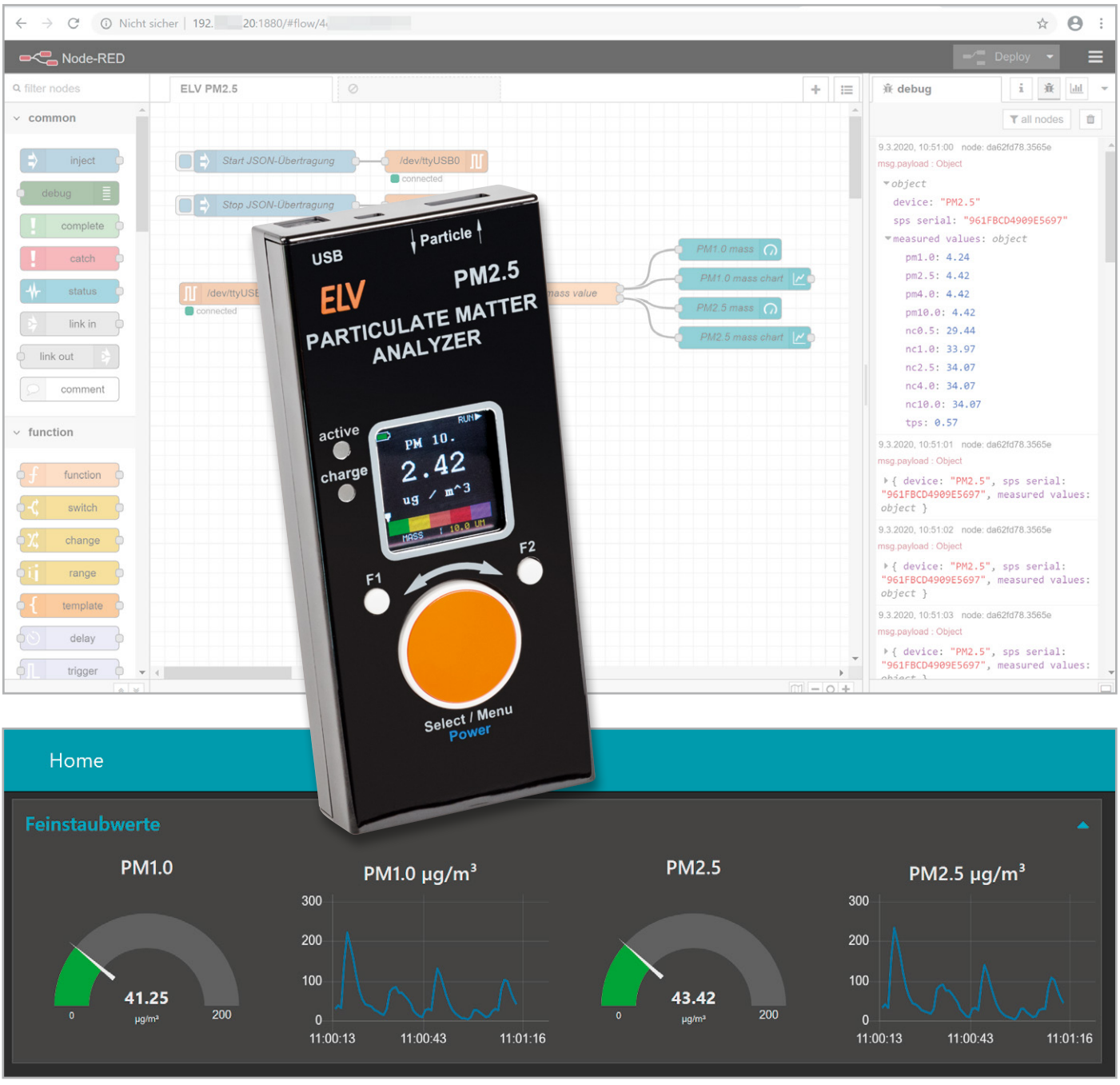
Dashboard für Feinstaubmessungen

Anzeige von Umweltdaten mit dem Raspberry Pi und Node-RED

Mit unserem ELV Komplettbausatz Feinstaub-Messgerät PM2.5 haben wir kürzlich ein hochpräzises Umweltmessgerät vorgestellt. Das mobile Messwerkzeug hat mit OLED-Display, Datenlogger und LiPo-Akku alle Voraussetzungen für Messungen im Feld. Aber auch kabelgebunden lässt sich das Messinstrument zur Auswertung nutzen. Im folgenden Beitrag zeigen wir aufbauend auf der Einführung in Node-RED in diesem ELVjournal (siehe Seite 24), wie man mithilfe von einfachen Software-Tools eine schicke Oberfläche dafür entwickeln kann. Natürlich kann man diesen Ansatz auch für alle anderen Geräte mit der Ausgabe von seriellen Daten im JSON-Format nutzen.



Allgemeines

Wie schädlich eine hohe Feinstaubbelastung für den Menschen ist, haben wir im ELVjournal 6/2019 [1] ausführlich erläutert. Unser ELV Komplettbausatz Feinstaub-Messgerät PM2.5 [2] kann Feinstaub mit einer Partikelkonzentration bis hinab zu PM1.0 und damit die für den Menschen unter Umständen sehr gefährlichen Kleinstpartikel messen. Der im Messgerät verbaute hochpräzise Sensor des Schweizer Sensorherstellers Sensirion hat zudem vor Kurzem als erster Massenmarkt-Feinstaubsensor eine MCERTS-Zertifizierung erhalten [3]. Aufgrund der Nachfrage ist das Messinstrument nun auch als Fertiggerät bei ELV verfügbar [2]. Das Messgerät kann sowohl mobil als auch stationär eingesetzt werden. Mit einem Raspberry Pi und dem Prototyping-Tool Node-RED (siehe auch Node-RED-Einführung im Beitrag auf Seite 24) lässt sich für den stationären Einsatz eine schicke Anzeige-Oberfläche entwickeln.

Serielle Daten

Als Grundlage für unsere Anzeige-Oberfläche dient die serielle Ausgabe der Daten über die USB-Schnittstelle des Geräts. Bei der Entwicklung des Feinstaub-Messgeräts haben wir bereits die Möglichkeit eingebaut, über serielle Befehle mit dem Gerät zu kommunizieren. Mit den Schnittstelleneinstellungen (Tabelle 1) kann man mit einem einfachen Terminal-Programm wie Tera Term [4] bereits die verschiedenen Optionen ausprobieren (Bild 1).

Will man beispielsweise die aktuelle Firmware-Version auslesen, so reicht die Eingabe eines „V" im Terminal aus, um sie von dem Gerät zu erhalten (Bild 2). Will man die aktuellen Messdaten per Ausgabe im JSON-Format erzeugen, reicht ein „J" zum Auslösen der Ausgabe über die serielle Schnittstelle. Mit „E" beendet man die Übertragung der aktuell gemessenen Daten (Bild 3). Einen kompletten Überblick über alle möglichen Befehle zeigt Tabelle 2.

Ausgabe im JSON-Format

Die Ausgabe im Terminalprogramm ist zum schnellen Testen geeignet, nicht aber, wenn man das Feinstaub-Messgerät komfortabel bedienen oder mit einer Oberfläche den Status der Feinstaubkonzentration darstellen möchte. Mit den seriellen Daten im weit verbreiteten JSON-Format [5] kann man sie mit zahlreichen Software-Tools auslesen, zerlegen und weiterverarbeiten.

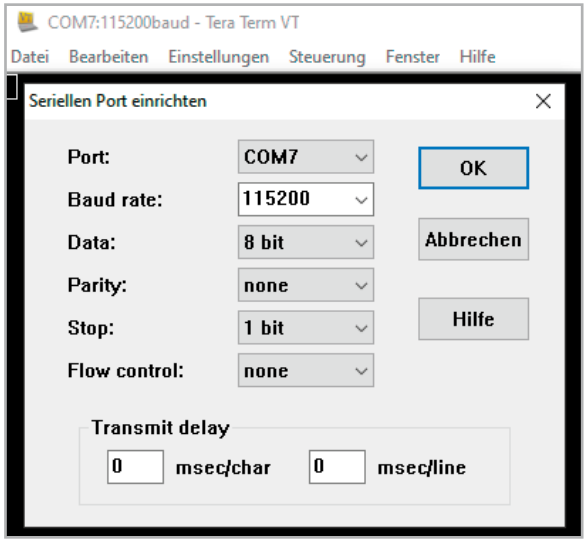


Bild 1: Schnittstelleneinstellungen für die serielle Kommunikation

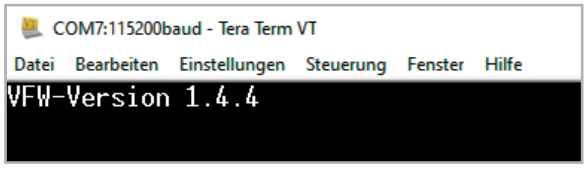


Bild 2: Auslesen der aktuellen Firmware-Version

Schnittstelleneinstellungen:	
Baudrate:	115.200 kbit/s
Datenbytes:	8
Parität:	keine
Stoppzeichen:	1

Tabelle 1

Befehlstabelle:

Befehlscode	Beschreibung
S	Startbefehl für die Liveübertragung im eingestellten Messraster
J	Startbefehl für die Liveübertragung im „Json"-Format (im eingestellten Messraster)
E	Übertragung der aktuell gemessenen Daten stoppen
X	Löschen aller aufgezeichneten Messdaten inkl. Reset aller Geräteeinstellungen
D	Löschen aller aufgezeichneten Messdaten
L	Anzahl der aufgezeichneten Messwerte im Gerät auslesen
R	Auslesen aller aufgezeichneten Messwerte
!	Starten des Geräte-Bootloaders
V	Auslesen der Firmware-Version
N	Auslesen der Sensorseriennummer des SPS30
A	Auslesen des Intervalls für automatische Sensorselbstreinigung

Tabelle 2

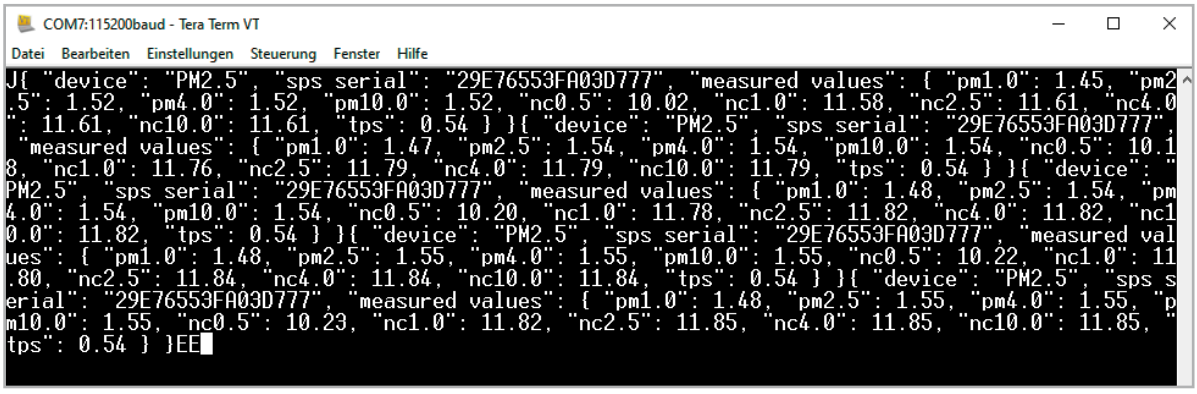


Bild 3: Ausgabe der Messdaten im JSON-Format

Typisch für JSON (JavaScript Object Notation) ist das kompakte Datenformat, das sich in einem einfach lesbaren Textformat zum Datenaustausch zwischen Anwendungen eignet. Die einzelnen Datensätze sind in Schlüssel-Werte-Paaren (key-value-pairs) gruppiert und durch einen Doppelpunkt getrennt und können zudem verschachtelt werden. Für die Werte gelten bestimmte Datentypen als Voraussetzung. In Tabelle 3 ist ein Beispiel für die Ausgabe der Messwerte im JSON-Format zu sehen.

Will man beispielsweise aus einer Software heraus den Wert für den Schlüssel „device" abrufen, so erhält man als Ausgabe „PM2.5", d. h., dem Schlüssel ist nur ein einziger Wert zugeordnet. Anders ist es beim Schlüssel „measured values", der ohne weitere genaue Spezifizierung alle aufgeführten Schlüsselwerte-Paare ausgeben würde. Hier kann man sich in der auszulesenden Software einzelne Werte zu Schlüsseln ausgeben lassen, also z. B. zu measured values nur den Wert zum Schlüssel „pm1.0". Wie das genau funktioniert, sehen wir in dem folgenden Programmierbeispiel für unsere Node-RED-Oberfläche.

Beispiel für eine Ausgabe der Messwerte im „Json"-Format per Liveübertragung:

Ausgabe	Beschreibung	Einheit
{		
"device": "PM2.5",	Gerätename	
"sps serial": "C0A012F92339FEDF",	Sensorseriennummer des SPS30	
"measured values":	Messwerte	
{		
"pm1.0": 4.23,	Massenkonzentration PM1.0	µg/m³
"pm2.5": 4.42,	Massenkonzentration PM2.5	µg/m³
"pm4.0": 4.42,	Massenkonzentration PM4	µg/m³
"pm10.0": 4.42,	Massenkonzentration PM10	µg/m³
"nc0.5": 29.19,	Partikelkonzentration PM0.5	#/cm³
"nc1.0": 33.71,	Partikelkonzentration PM1.0	#/cm³
"nc2.5": 33.82,	Partikelkonzentration PM2.5	#/cm³
"nc4.0": 33.82,	Partikelkonzentration PM4	#/cm³
"nc10.0": 33.82,	Partikelkonzentration PM10	#/cm³
"tps": 0.47	Typische Partikelgröße	µm
}		
}		

Tabelle 3

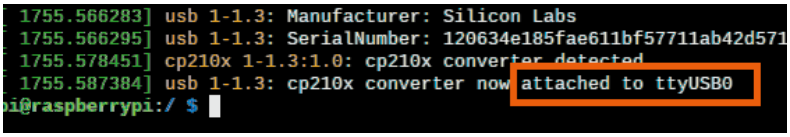


Bild 4: Per dmesg können die Details für den USB-Port angezeigt werden.

Unterbau für die Oberfläche

Für die Programmierung in Node-RED müssen wir uns zunächst darum kümmern, den USB-Anschluss für den angeschlossenen Feinstaubsensor festzustellen. Dazu öffnen wir ein Terminal auf dem Linux-Desktop (Strg-T) und schließen das Feinstaub-Messgerät mit einem USB-Kabel an den Raspberry Pi an. Mit dem Befehl dmesg schauen wir uns nun die zuletzt generierten Einträge im Nachrichtenpuffer des Linux-Kernels an, der u. a. Meldungen von Gerätetreibern auflistet (Bild 4).

Unser Feinstaub-Messgerät ist an der seriellen Schnittstelle ttyUSB0 angeschlossen, was wir uns merken. Nachdem wir unser Node-RED mit node-red-start gestartet haben, öffnen wir im Browser (http://IP_Raspberry_Pi:1880) den Node-RED-Editor (siehe Node-RED-Einführung auf Seite 24) und fügen dem Flow einen Serial-out-Node hinzu. Diesen Node brauchen wir für alle Befehle, die wir an das Feinstaub-Messgerät senden wollen, also beispielsweise um die Übertragung der JSON-Daten zu starten oder wieder zu stoppen.

Wir doppelklicken als Nächstes auf den Serial-out-Node und im öffnenden Fenster auf das Stiftsymbol neben „Serial Port". Im nächsten Fenster „Properties" können wir dann die anzupassenden Schnittstelleneinstellungen hinzufügen (Bild 5):

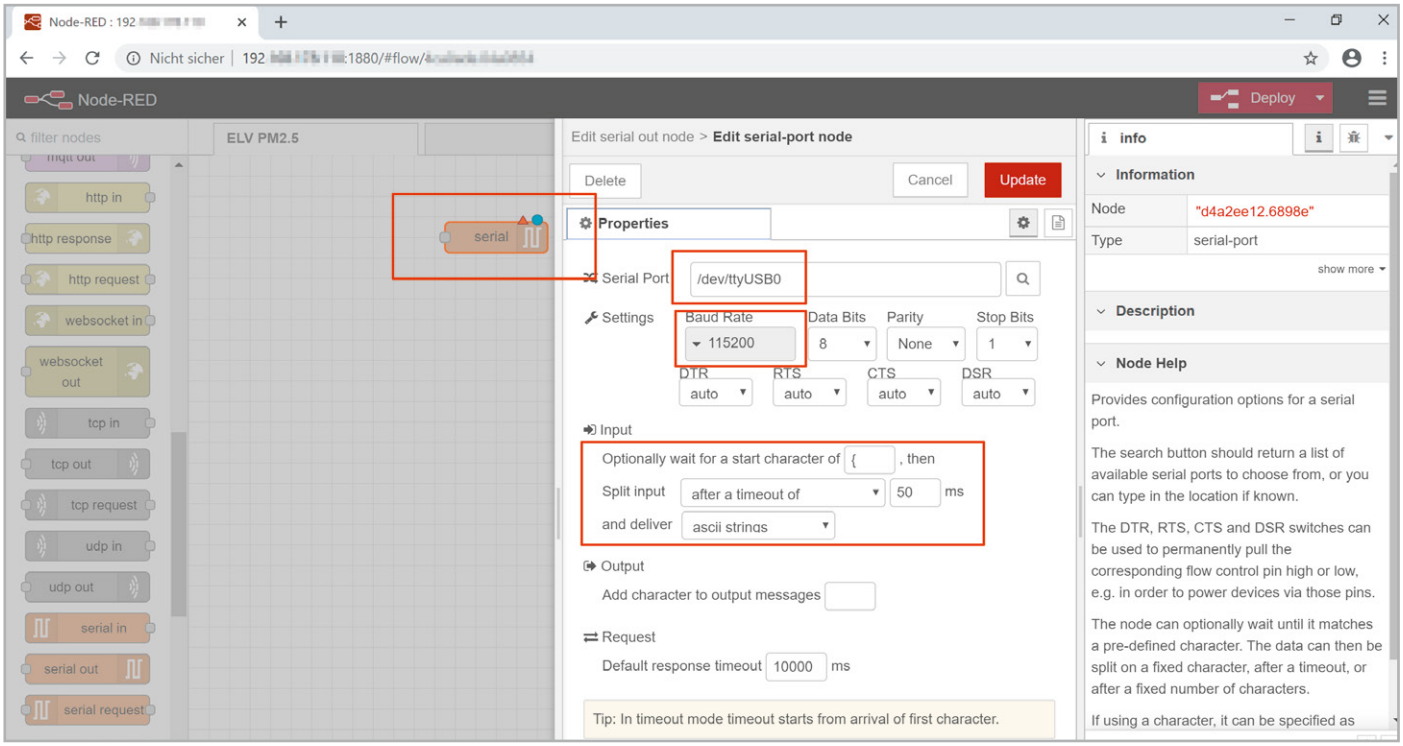


Bild 5: Konfiguration der Schnittstelleneinstellungen

Serial Port:	wird aus der dmesg-Ausgabe bestimmt
Input:	
Baudrate:	115200
Optionally wait for a start character of:	{
Split input:	after a timeout of 50 ms

Anschließend klicken wir rechts oben auf „Update“ und im folgenden Fenster auf „Done“. Der Serial-out-Node ist nun konfiguriert. Um die Übertragung der JSON-Daten zu starten benötigen wir einen Inject-Node, den wir links neben den Serial-out-Node ziehen. Dort definieren wir einen String als Payload und fügen ein großes „J“ ein. Zusätzlich geben wir dem Inject-Node noch einen Namen – beispielsweise „Start JSON-Übertragung“ (Bild 6).

Wir klicken auf „Done“, verbinden beide Nodes und klicken auf „Deploy“, um den Flow zu aktivieren. Am Serial-out-Node sollte ein grünes Quadrat und der Status als „connected“ erscheinen. Ist dies nicht der Fall, sollte kontrolliert werden, ob wirklich die richtige serielle Schnittstelle bzw. die korrekten Einstellungen eingegeben wurden.

Wir kopieren die beiden Nodes, indem wir mit der Maus einen Kasten um sie ziehen und mit Strg+C kopieren und mit Strg-V neu einfügen. Bei dem neu erzeugten Node-Pärchen verändern wir nur den Inject-Node und fügen statt „J“ ein „E“ ein und ändern den Namen auf „Stop JSON-Übertragung“. Mit einem Klick auf das linke Fähnchen der jeweiligen inject Nodes können wir nun zukünftig die JSON-Übertragung starten und stoppen.

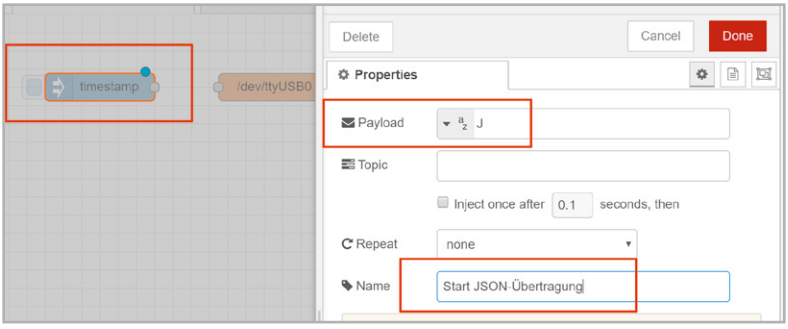


Bild 6: Konfiguration des Inject-Nodes

Ausgabe von Debug-Messages

Nun wollen wir kontrollieren, ob wir auch tatsächlich JSON-Daten von unserem Feinstaub-Messgerät erhalten. Dazu wählen wir als Erstes einen Serial-in-Node aus und konfigurieren diesen analog zu den Serial-out-Nodes. An diesen Serial-in-Node hängen wir einen Debug-Node, verbinden beide und klicken auf „Deploy“. Im rechten Fenster wählen wir die Anzeige der Debug-Messages per Klick auf das Käfer-Symbol. Klicken wir nun auf das linke Fähnchen des Inject-Nodes „Start JSON-Übertragung“, sollte unmittelbar der Datenstrom im Debug-Fenster starten (Bild 7).

Die Daten sehen zwar schon nach dem JSON-Format aus, wir können in dieser Form aber im Nachrichtenstrom von Node-RED noch nicht so elegant zugreifen, wie das möglich wäre. Deswegen fügen wir zwischen Serial-in-Node und Debug-Node noch einen JSON-Node ein, stoppen den JSON-Datenstrom, deployen das ganze und starten die Datenübertragung wieder.

Aus dem String ist nun ein Objekt geworden, bei dem wir gezielt auf die Werte des zugehörigen Schlüssels zugreifen können. Verändern wir beispielsweise in dem Debug-Node den Output von msg.payload auf msg.payload.device erhalten wir den Wert für den Schlüssel „device“ – in diesem Fall also den String „PM2.5“. In dieser Weise ist jetzt der Zugriff auf alle Werte möglich. Für die Ausgabe des aktuellen Wertes für die Partikelkonzentration PM2.5 müssten wir den Output auf msg. payload["measured values"]["pm2.5"] ändern.

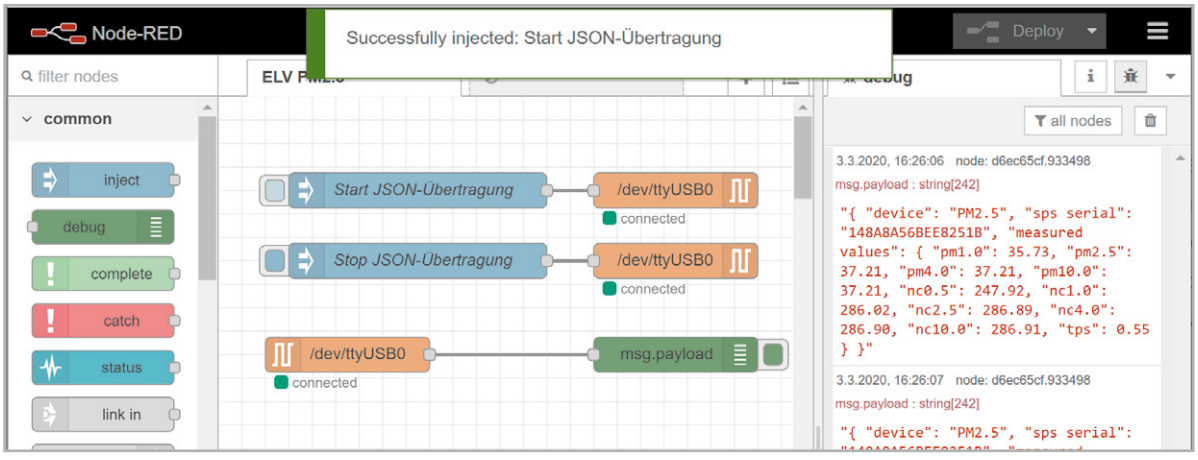


Bild 7: Datenausgabe im Debug-Fenster

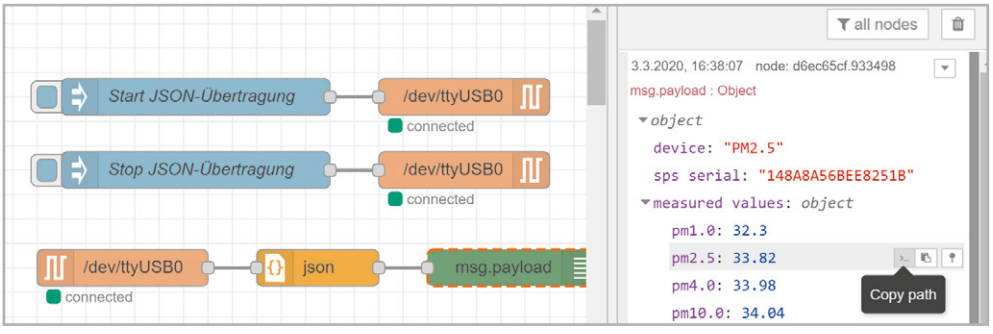


Bild 8: „Copy path“ ist sehr hilfreich, vor allem bei längeren Payload-Konstrukten.

Tipp: Wenn man das JSON-Objekt im Debug-Fenster erweitert und auf das linke der drei kleinen Symbole neben dem jeweiligen Schlüssel-Werte-Paar klickt („Copy path"), erhält man die korrekte Definition für die Payload (Bild 8).

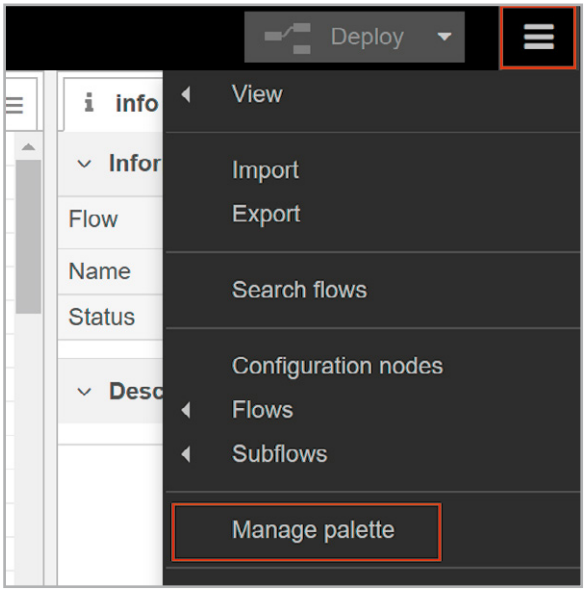


Bild 9: Die Installation von zusätzlichen Paketen erfolgt über „Manage palette".

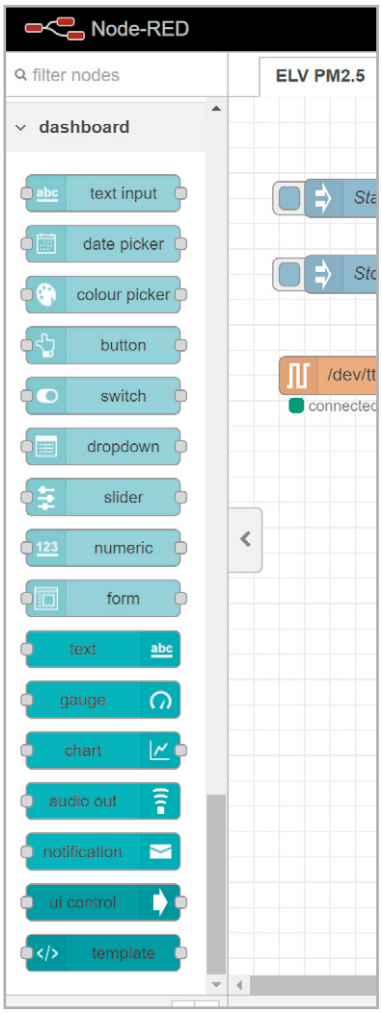


Bild 10: Neu vorhandene Nodes aus dem Dashboard-Paket

Wir können bis hierher den JSON-Datenstrom starten, stoppen und uns per Debug-Nachrichten Daten anzeigen lassen. Das ist nur eine rudimentäre Lösung und nicht besonders gut ablesbar, geschweige denn schön anzusehen. Node-RED bringt dafür mit dem Dashboard-Node ein Paket mit vielen Anzeigemöglichkeiten in einer ansehnlichen Oberfläche mit.

Um das Paket zu installieren, geht man im „Hamburger"-Menü rechts oben auf „Manage palette" (Bild 9) und gibt im sich öffnenden Fenster unter dem Reiter „Install" den Begriff „dashboard" ein. Hier wählt man „node-red-dashboard" zur Installation und bestätigt mit „Install" die nachfolgenden Dialoge.

Nach der Installation erscheinen im linken Fenster die neuen für das Dashboard verfügbaren Nodes (Bild 10). Als Vorbereitung für die spätere Ausgabe in unserem Dashboard-Element „Gauge" (eine Art Messuhr) suchen wir uns zunächst das Schlüssel-Werte-Paar heraus, dessen Wert wir anzeigen lassen wollen.

Für die Ausgabe des aktuellen Wertes für die Partikelkonzentration PM2.5 müssen wir den Output auf msg. payload["measured values"]["pm2.5"] (s. o.) setzen. Da Node-RED ein flussbasiertes Nachrichtensystem ist, wird von jedem Node unter anderem die Nachricht msg.payload weitergereicht.

Unser bisheriger Flow beinhaltet damit das gesamte JSON-Objekt. Um die msg.payload Nachricht anzupassen und nur den Wert zu dem Schlüssel „pm2.5" anzeigen zu lassen, ziehen wir einen Change-Node in den Editor und setzen folgende Einstellungen (Bild 11):

Move: msg.payload["measured values"]["pm2.5"]
To: msg.payload

Tipp: Man kann hierfür natürlich auch einen Function-Node verwenden. Der verwendete Weg soll aber gänzlich ohne Programmierung im eigentlichen Sinne auskommen. Analog zu unserem Change-Node würde der Code in der Function-Node folgendermaßen aussehen:

```
msg1 = {};
msg1.payload = msg.payload["measured values"]["pm1.0"];
return msg1;
```

Entsprechend könnte man so gleichzeitig mehrere Werte aus den JSON-Daten auslesen. Dabei muss man beachten, dass die Rückgabewerte in eine eckige Klammer gesetzt werden und der Function-Node mit entsprechend vielen Ausgängen versehen werden muss:

```
return [msg1, msg2, msg3];
```

Die Nachricht msg.payload beinhaltet damit nur noch den Wert für die Feinstaubkonzentration PM2.5 und kann so an den folgenden Node – den Dashboard-Gauge-Node, den wir als Nächstes in den Editor ziehen – übergeben werden.

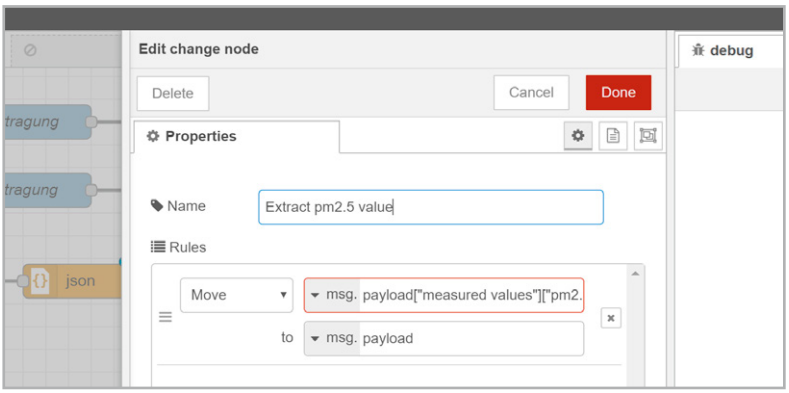


Bild 11: Anpassen der Nachricht auf die Ausgabe des PM2.5-Wertes

In dem Gauge-Node setzen wir folgende Einstellungen für die

Anzeige: Add new ui_group…
(auf das Stift-Icon klicken)

Im folgenden Fenster:

Name: Feinstaubwerte
Tab: Add new ui_tab
(auf das Stift-Icon klicken)

Im folgenden Fenster die Einstellungen so belassen und dann jeweils mit Done/Update bestätigen.

Nun müssen wir noch die Einstellungen für die Anzeige der Messuhr (gauge) definieren:

Label: PM2.5 Konzentration
Units: µg/m³
Range: min 0 max 200
Sectors 0 50 100 200
Name: PM2.5

Die Einstellungen sollten dann wie in Bild 12 aussehen, werden mit Done bestätigt und der gesamte Flow deployed.

Das Dashboard rufen wir nun im Browser auf per http://IP_Raspberry_Pi:1880/ui/ und es erscheint die Anzeige wie in Bild 13.

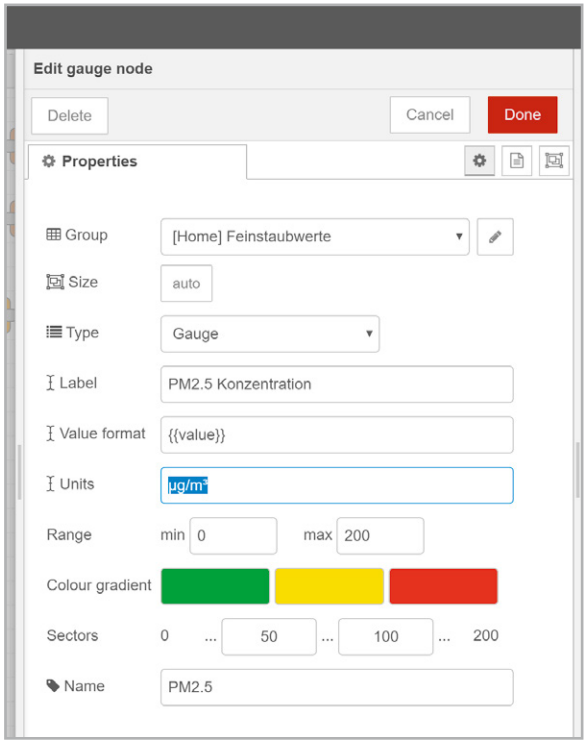


Bild 12: Einstellungen für den Gauge-Node

Ausblick

Wir haben in diesem Beitrag nur an der Oberfläche der Möglichkeiten gekratzt, die sich mit der Auswertung von Messdaten und der Anzeige mit einer visuellen Programmierumgebung befassen. Das Dashboard kann man mit verschiedenen Ausgaben beliebig ergänzen und so z. B. alle Werte gleichzeitig in einer Messuhr, per Text oder in einem Chart darstellen. Zudem kann man die Daten mit einfachen Textdateien oder Datenbanken-Nodes persistieren, um so bei einem notwendigen Node-RED-Neustart die gespeicherten Werte zu laden. Will man auf die Werte von außen zugreifen, bietet sich zum einen die Verbindung per VPN auf den lokalen Raspberry Pi an oder, soweit man dem Versand der Daten an externe Server vertraut, zum anderen auch das Versenden an Plattformen wie ThingSpeak. Wie das funktioniert haben wir im ELVjournal 2/2020 [6] erklärt.

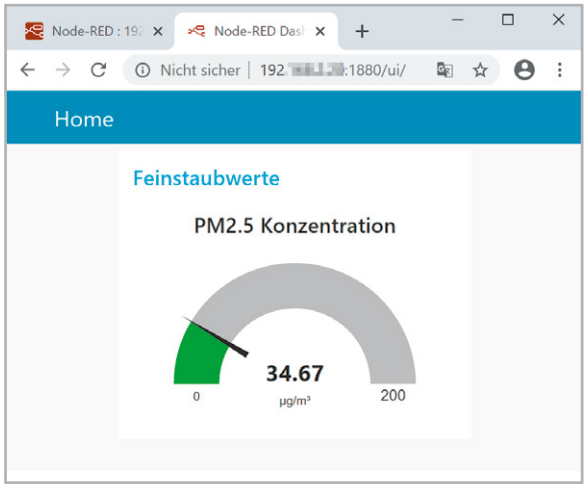


Bild 13: Anzeige der Dashboard-Messuhr

Weitere Infos:

[1] Verschmutzte Luft: Feinstaub – Genaue Messungen mit dem ELV Feinstaub-Messgerät PM2.5
de.elv.com: Bestell-Nr. 251073 (Fachbeitrag)
[2] Feinstaub-Messgerät PM2.5
ELV Komplettbausatz: Bestell-Nr. 154618
ELV Fertiggerät: Bestell-Nr. 155460
[3] Erster Massenmarkt-Feinstaubsensor erhält MCERTS-Zertifizierung:
www.sensirion.com/de/ueber-uns/newsroom/news-und-pressemitteilungen/detail/news/erster-massenmarkt-feinstaubsensor-erhaelt-mcerts-zertifizierung/
[4] Tera Term: https://ttssh2.osdn.jp/
[5] JSON: de.wikipedia.org/wiki/JavaScript_Object_Notation
[6] WLAN Fernsteuerung – ESP32 – WLAN und Webserver in MicroPython
de.elv.com: Bestell-Nr. 251238 (Fachbeitrag)

Alle Links finden Sie auch online unter de.elv.com/elvjournal-links