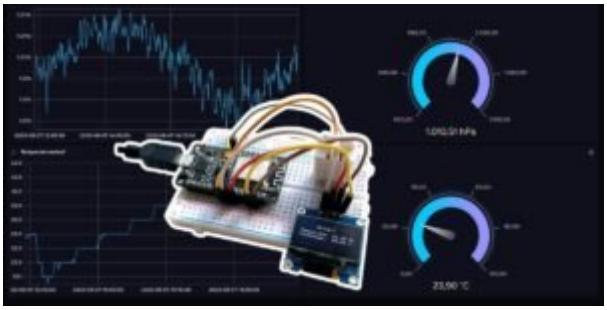
ESP8266 Wetterstation mit Datenaufzeichnung und -visualisierung



Baue eine ESP8266 Wetterstation, die dir die aktuelle Temperatur, Luftfeuchtigkeit und den Luftdruck anzeigt sowie deine Daten speichert und visualisiert. Die aktuellen Messdaten erscheinen auf einem kleinen OLED-Display. **Aber das ist nicht alles: Deine Messdaten speicherst du in einer Datenbank, um auf vergangene Messungen zugreifen und sie auswerten zu können.**

Um deine Messdaten zu speichern, verwendest du die **Datenbank InfluxDB**, die sicher hervorragend dazu eignet, zeitgebundene Daten zu speichern und nebenbei auch noch die Visualisierung deiner Daten einfach und intuitiv ermöglicht. **InfluxDB wird lokal auf einem Raspberry Pi laufen.** Der ESP8266 sendet die Messdaten dorthin und InfluxDB speichert und visualisiert sie. Anzeigen lassen kannst du diese dir dann in einem Browser und auf einem Gerät deiner Wahl.

Die Software, die du für die ESP8266 Wetterstation benötigst, ist kostenlos verfügbar.

Inhalte dieses Projekts:

- Die Sensoren DHT22 und BMP180 am ESP8266 anschließen und verwenden
- Messdaten auf einem OLED-Display anzeigen

- Den Raspberry Pi einrichten
- Per SSH auf deinen Raspberry Pi zugreifen
- InfluxDB installieren und die Messdaten vom ESP8266 übertragen
- Auf die Daten zugreifen und sie anschaulich visualisieren

Diese Bauteile benötigst du für die ESP8266 Wetterstation:

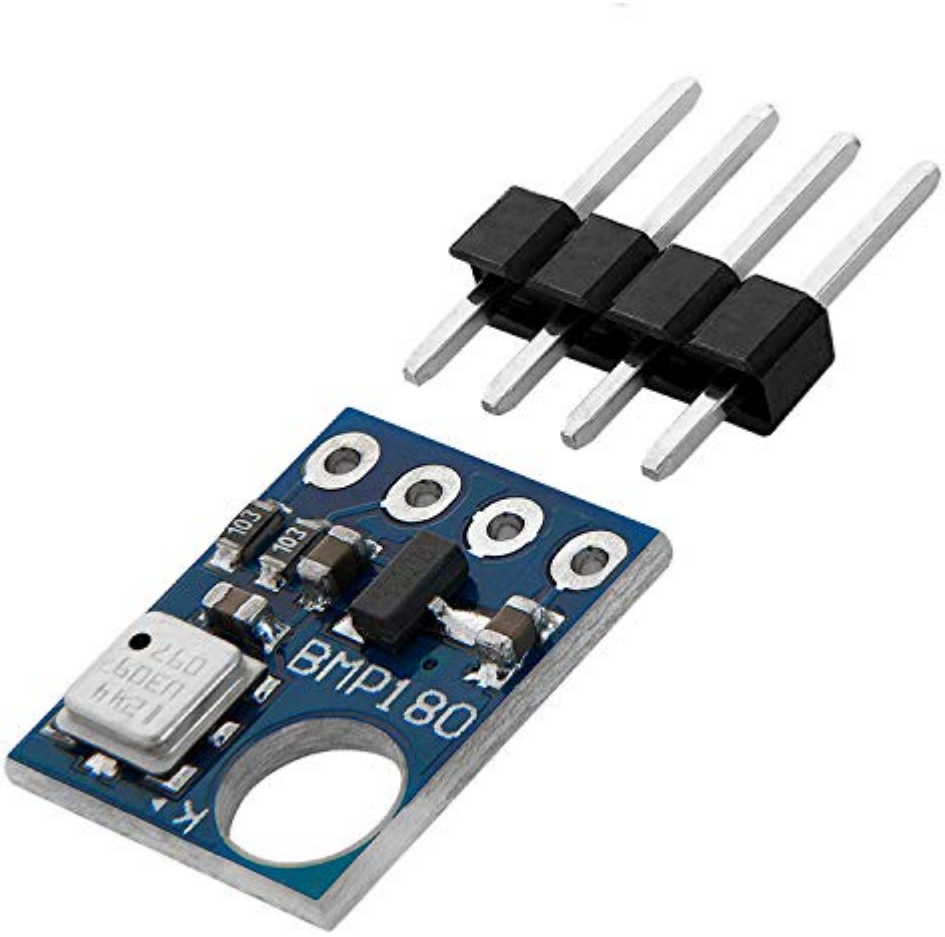
AZDelivery NodeMCU Amica Modul V2 ESP8266 ESP-12F WiFi - Node MCU ESP 8266 WiFi Development Board mit CP2102 kompatibel mit Arduino - inklusive Installationsanleitung als E-Book
 Maße (LxBxH): 48 x 26 x 13 mm
8,49 €

AZDelivery DHT22 AM2302 Temperatursensor und Luftfeuchtigkeitssensor kompatibel mit Arduino und Raspberry Pi inklusive E-Book!

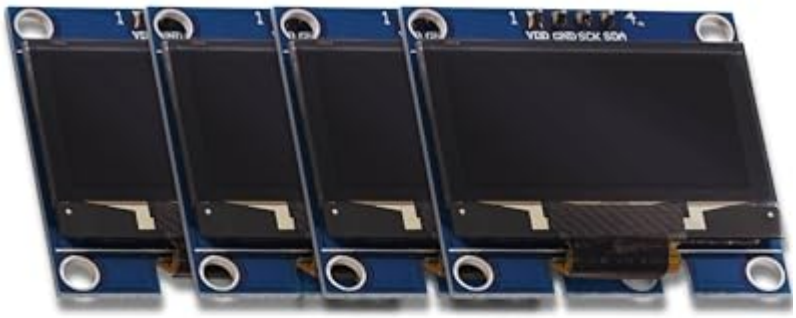
 Neben der Betriebsspannung muss lediglich ein Port mit dem Sensor verbunden werden.

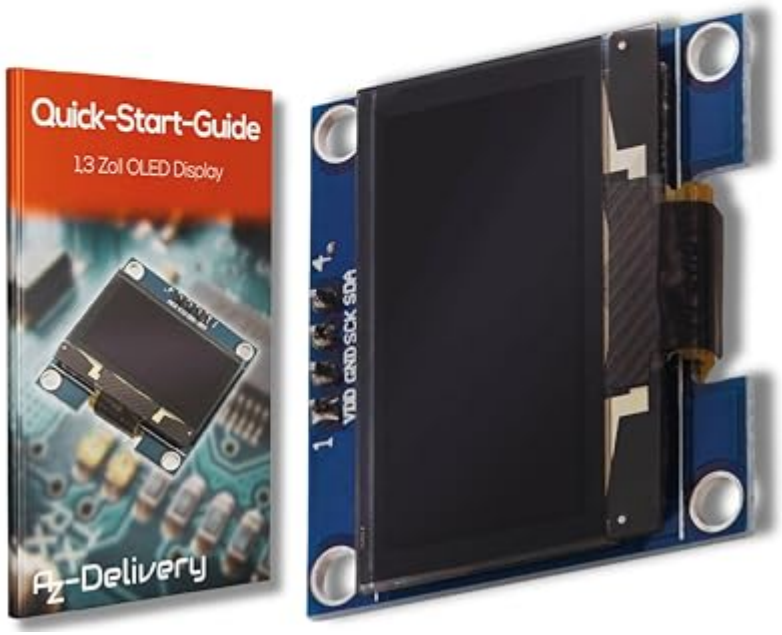
9,49 €

Angebot



[AZDelivery GY-68 BMP180 Barometrischer Luftdruck und Temperatur Sensor kompatibel mit Arduino und Raspberry Pi inklusive E-Book!]
4,49 €





[AZDelivery 1 x 1,3 Zoll OLED Display I2C SSH1106 Chip 128 x 64 Pixel I2C Bildschirm Anzeigemodul mit weißen Zeichen | I2C Display ssd1306 | kompatibel mit Arduino und Raspberry Pi inklusive E-Book!]

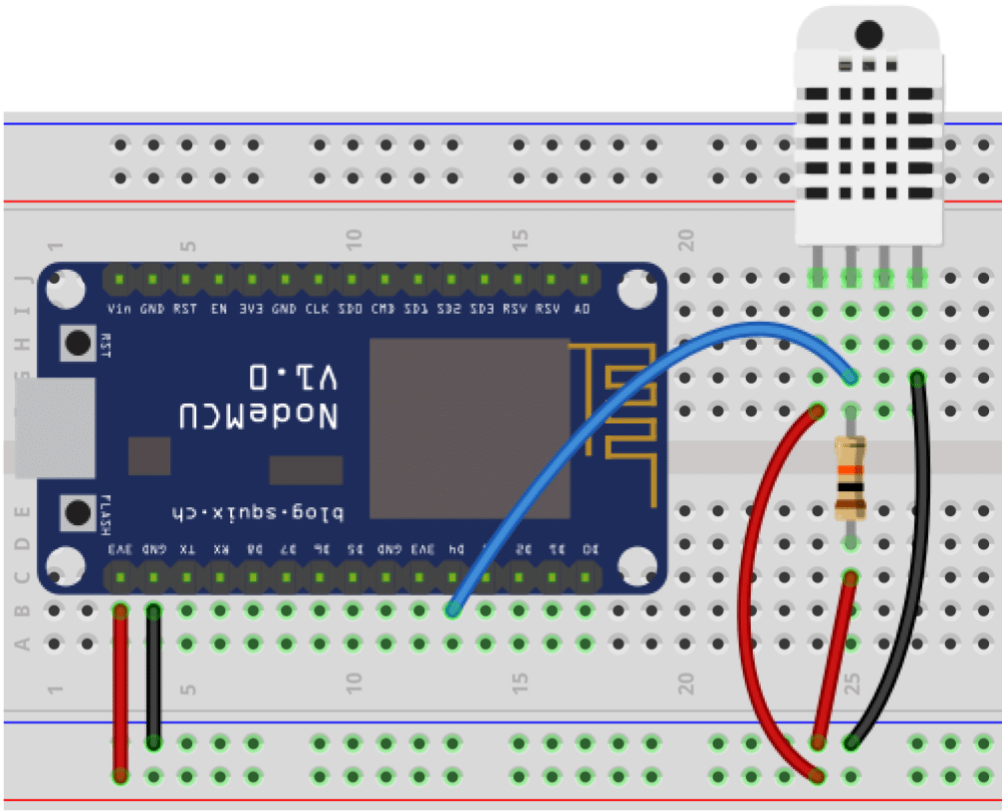
8,49 €

Angebot

[Raspberry Pi 4 Modelo B 4GB]
70,99 €

Den DHT22 AM ESP8266 anschließen und verwenden

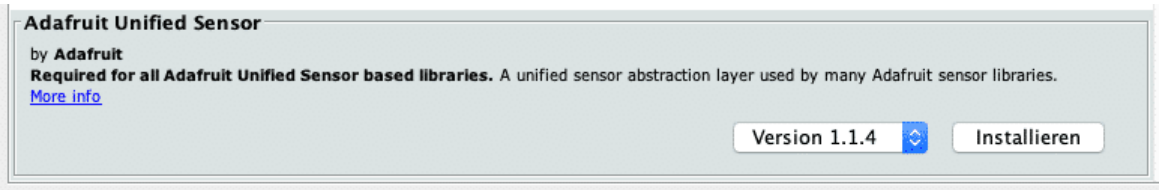
Der Temperatursensor DHT22 misst neben der Temperatur auch die Luftfeuchtigkeit. Um ihn an deinem ESP8266 anzuschließen, orientiere dich an der folgenden Skizze. Achte bitte auf den 10 kΩ Widerstand, den du zwischen dem Anschluss des DHT22 am Pin D4 des ESP8266 und Plus einsetzen musst.



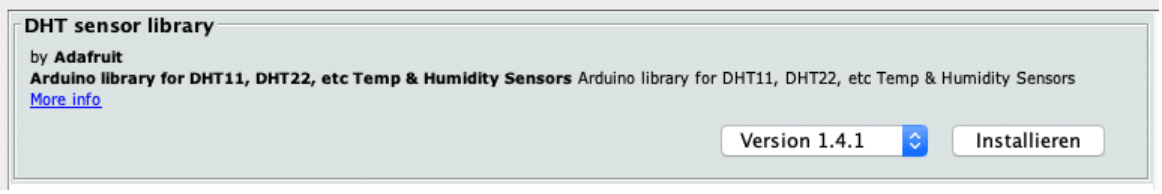
Übrigens: Falls du deinen ESP8266 noch nicht in der Arduino IDE verfügbar gemacht hast, findest du hier bei uns ein passendes Tutorial.

DIE PASSENDEN BIBLIOTHEKEN

Um deinen Sensor verwenden zu können, musst du zwei Bibliotheken installieren, von denen du jedoch nur eine im Sketch einbinden musst. Öffne deinen Bibliotheksmanager. Suche dort zunächst nach **Adafruit Unified Sensor** und installiere die aktuelle Version. Die Versionsnummern in den folgenden Screenshots können abweichen.



Suche anschließend nach **DHT sensor library** und installiere die entsprechende Bibliothek.



DIE TEMPERATUR und Luftfeuchtigkeit MESSEN

Kopiere dir den folgenden Sketch und lade ihn auf deinen Arduino hoch:

```
#include "DHT.h"

#define DHTPIN D4
#define DHTTYPE DHT22

float tempDHT22;
float humidity;

DHT dht(DHTPIN, DHTTYPE);
```

```
void setup() {
  Serial.begin(115200);
  dht.begin();
}

void loop() {

  tempDHT22 = dht.readTemperature();
  humidity = dht.readHumidity();

  Serial.print("Temperatur: ");
  Serial.print(tempDHT22);
  Serial.println("*C");
  Serial.print("Luftfeuchtigkeit: ");
  Serial.print(humidity);
  Serial.println("%");
  Serial.println();
  delay(2000);
}
```

So funktioniert der Sketch: Nachdem du die Bibliothek eingebunden hast, legst du den Pin fest, an dem der Sensor angeschlossen ist. In unserem Sketch ist das der Pin **D4**.

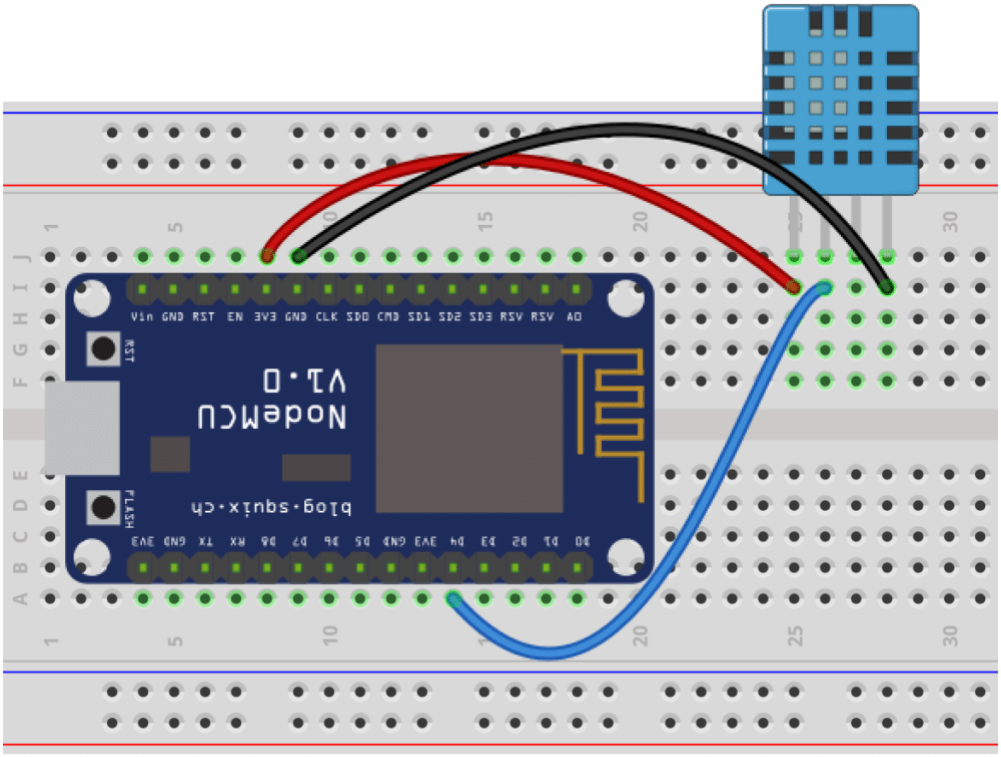
In der nächsten Zeile legst du das Modell des Sensors fest – in unserem Fall also ein DHT22. Anschließend erstellst du ein Objekt der Bibliothek names **dht**, das später bei der Messung mit den Funktionen **dht.readTemperature()** und **dht.readHumidity()** zum Einsatz kommt.

Der Rest des Sketchs dürfte für dich kein Problem sein. **Achte jedoch darauf, dass die Baudrate von Sketch und Seriellem Monitor übereinstimmt.** Hinweis: Es kann auch sein, dass dein DHT22 auch ohne Widerstand funktioniert – oder auch nur ohne Widerstand.

Wenn du doch lieber den Sensor DHT11 verwenden möchtest, musst du nur eine Stelle im Sketch anpassen:

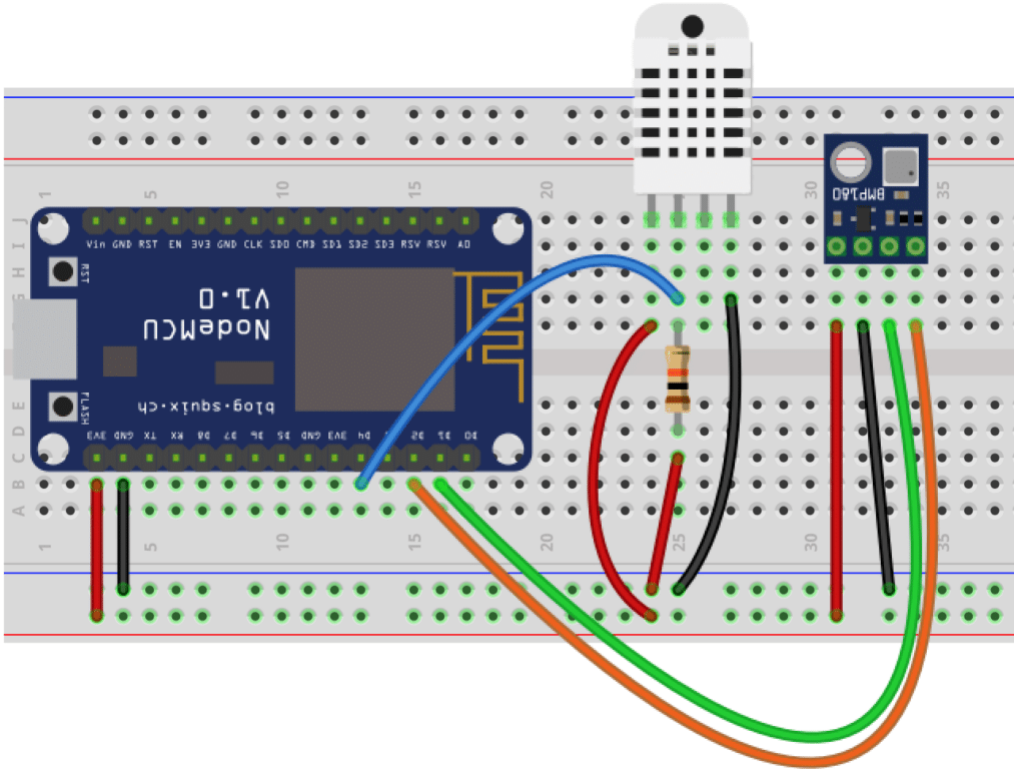
```
#define DHTTYPE DHT11
```

Wie du den „kleinen Bruder“ des DHT22 – also den DHT11 – anschließt, erfährst du in der folgenden Skizze. Im weiteren Verlauf des Projekts verwenden wir jedoch weiterhin den DHT22.



DEN BMP180 am ESP8266 anschließen und verwenden

Neben der Luftfeuchtigkeit und der Temperatur soll die ESP8266 Wetterstation auch den aktuellen Luftdruck messen. Hierfür eignet sich der Sensor BMP180. Da dieser auch die Temperatur messen kann, schauen wir uns gleich auch noch die Messunterschiede zwischen DHT11 und BMP180 an. Doch zunächst zum Anschluss – orientiere dich hierbei an der folgenden Skizze:



Die benötigte Bibliothek

Neben der bereits vorinstallierten Bibliothek **Wire** (für die Kommunikation per I²C), benötigst du noch eine weitere, um die Daten des Sensors problemlos auslesen zu können.

Öffne also den Bibliotheksmanager in der Arduino IDE und suche nach **BMP180**. Du findest nun eine Bibliothek namens **Adafruit BMP085 Library** – das ist die richtige, auch wenn sie ein anderes Modell im Namen trägt. Der BMP085 war das Vorgängermodell des BMP180, was die Kommunikation angeht, jedoch mehr oder weniger baugleich.

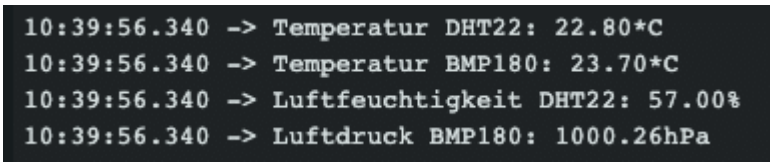
Die Temperatur und den Luftdruck messen

Nun erweiterst du den obigen Sketch um den Code für den BMP180:

```
#include "DHT.h"
#include "Wire.h"
#include "Adafruit_BMP085.h"

#define DHTPIN D4
#define DHTTYPE DHT22

Adafruit_BMP085 bmp;

float tempDHT22;
float tempBMP180;
float humidity;
float pressure;

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  dht.begin();

  if (!bmp.begin()) {
    Serial.println("Sensor BMP180 nicht gefunden!");
    while (1) {}
  }
}

void loop() {

  tempDHT22 = dht.readTemperature();
  tempBMP180 = bmp.readTemperature();
  humidity = dht.readHumidity();
  pressure = bmp.readPressure();

  Serial.print("Temperatur DHT22: ");
  Serial.print(tempDHT22);
  Serial.println("*C");
```

```
  Serial.print("Temperatur BMP180: ");
  Serial.print(tempBMP180);
  Serial.println("*C");

  Serial.print("Luftfeuchtigkeit DHT22: ");
  Serial.print(humidity);
  Serial.println("%");

  Serial.print("Luftdruck BMP180: ");
  Serial.print(pressure/100);
  Serial.println("hPa");

  Serial.println();
  delay(2000);
}
```

Sobald du den Sketch auf deinen ESP8266 geladen hast, sollten im seriellen Monitor die vier Messdaten erscheinen.



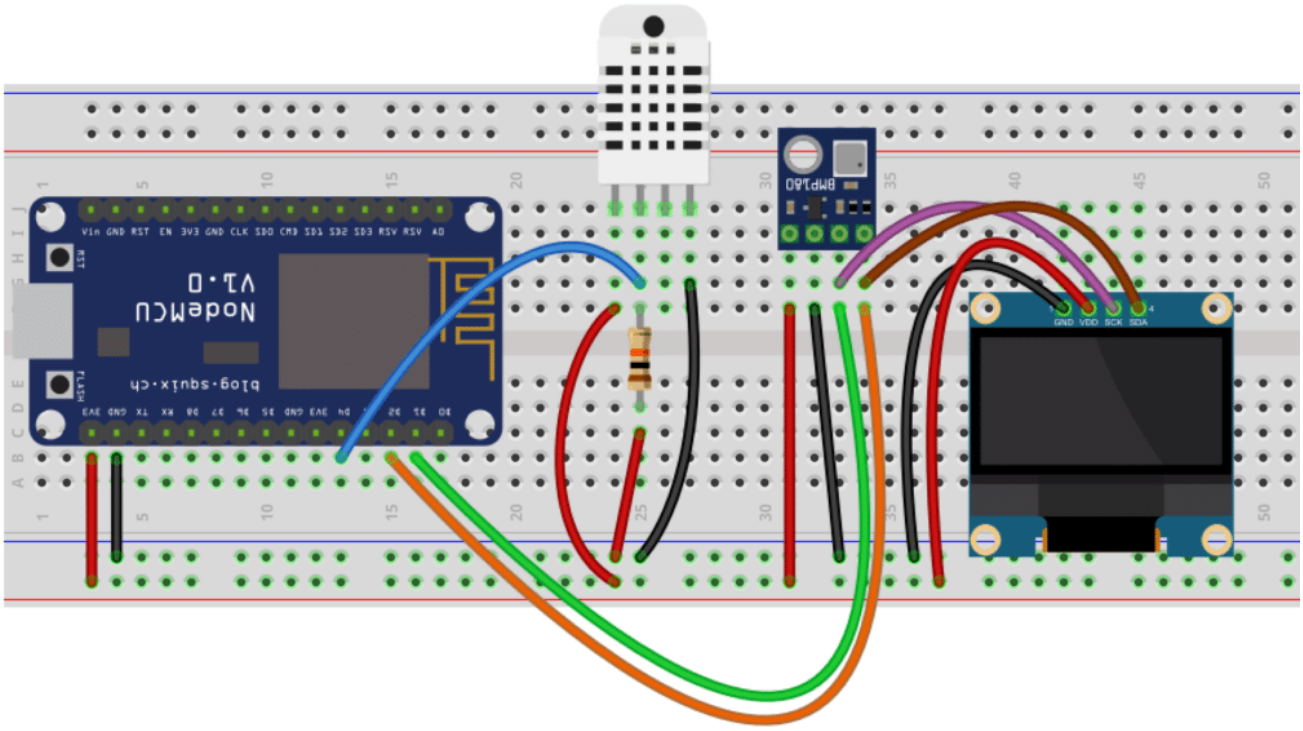
In den allermeisten Fällen dürfte die Temperaturmessung des DHT22 und des BMP180 etwas auseinander liegen. Ein Blick in die jeweiligen Datenblätter verrät, dass beide Sensoren eine Genauigkeit von ±0,5°C haben – das ist eigentlich schon recht genau und für eine Wetterstation sicherlich genau genug. Für welchen Messwert du dich entscheidest, liegt nun bei dir – vielleicht hast du noch ein weiteres Thermometer zur Hand, das du als Referenz einsetzen kannst.

Im weiteren Verlauf dieses Projekts verwenden wir die Temperaturdaten des BMP180.

Das OLED-Display anschließen

Aktuell siehst du deine Messdaten nur im seriellen Monitor. Deshalb kommt nun ein Display zum Einsatz, auf dem du sie bequemer ablesen kannst. In diesem Projekt verwenden wir das handelsübliche **OLED-Display Adafruit SSD1306 mit einer Größe von 128×64 px.**

Erweitere also den Aufbau auf deinem Breadboard wie folgt:



Wie du siehst, sind sowohl der BMP180 als auch das OLED-Display per I²C (also an den Pins D1 und D2) am ESP8266 angeschlossen. Damit der Microcontroller beide Bauteile ansprechen kann, besitzen sie unterschiedliche Adressen – das OLED-Display mit 128×64 px die Adresse **0x3C** und der BMP180 die Adresse **0x77**.

Die benötigten Bibliotheken

Nun ist deine ESP8266 Wetterstation vollständig. Allerdings fehlt noch der Sketch, mit dem du deine Messdaten auf dem

OLED-Display anzeigst. Auch für das Display benötigst du die Unterstützung von Bibliotheken, die du im Handumdrehen installierst. Öffne also wieder den Bibliotheksmanager und suche zunächst nach **Adafruit SSD1306** und installiere die neueste Version. Falls du gefragt wirst, ob du auch zugehörige Erweiterungen installieren möchtest, bestätige das mit einem Ja.

Die zweite Bibliothek findest du mit einer Suche nach **Adafruit GFX Library**. Falls diese schon im Zuge der ersten Installation mitinstalliert wurde, brauchst du nichts weiter zu tun und kannst den Bibliotheksmanager schließen.

Messdaten auf dem Display anzeigen

Ersetze den Code auf deinem ESP8266 durch den folgenden erweiterten Sketch:

```
#include "DHT.h"
#include "Wire.h"
#include "Adafruit_BMP085.h"
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define DHTPIN D4
#define DHTTYPE DHT22

Adafruit_BMP085 bmp;

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

float tempDHT22;
float tempBMP180;
float humidity;
float pressure;
```

```
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  dht.begin();

  if (!bmp.begin()) {
    Serial.println("Sensor BMP180 nicht gefunden!");
    while (1) {}
  }

   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {   //
Display-Addresse: 0x3C für Groesse 128x64px
    Serial.println(F("SSD1306 allocation failed"));
    for (;;)
      ;
  }
  display.setTextSize(1);       //Schriftgröße
  display.setTextColor(WHITE);  //Schriftfarbe


  display.clearDisplay();
  display.display();
}

void loop() {

  tempDHT22 = dht.readTemperature();
  tempBMP180 = bmp.readTemperature();
  humidity = dht.readHumidity();
  pressure = bmp.readPressure();

  Serial.print("Temperatur DHT22: ");
  Serial.print(tempDHT22);
  Serial.println("*C");

  Serial.print("Temperatur BMP180: ");
  Serial.print(tempBMP180);
  Serial.println("*C");

  Serial.print("Luftfeuchtigkeit DHT22: ");
```

```
  Serial.print(humidity);
  Serial.println("%");

  Serial.print("Luftdruck BMP180: ");
  Serial.print(pressure / 100);
  Serial.println("hPa");
  Serial.println();

  display.clearDisplay();

  display.setCursor(10, 10);
  display.println("Temp.:  " + String(tempBMP180) + " *C");

  display.setCursor(10, 29);
  display.println("Luftf.: " + String(humidity) + " %");
  display.setCursor(10, 48);

  display.println("Luftd.: " + String(pressure/100) + " hPa");

  display.display();
  delay(2000);
}
```

Auf deinem OLED-Display sollten nun untereinander die Werte für Temperatur, Luftfeuchtigkeit und Luftdruck zu sehen sein, die alle zwei Sekunden aktualisiert werden.

Die Daten auf dem Raspberry Pi speichern und visualisieren

Messdaten auf einem Display sind eine tolle Sache – aber wenn du die Werte deiner Wetterstation über längere Zeit speichern und sie als Graphen anzeigen möchtest, musst du ein paar Schritte weiter gehen.

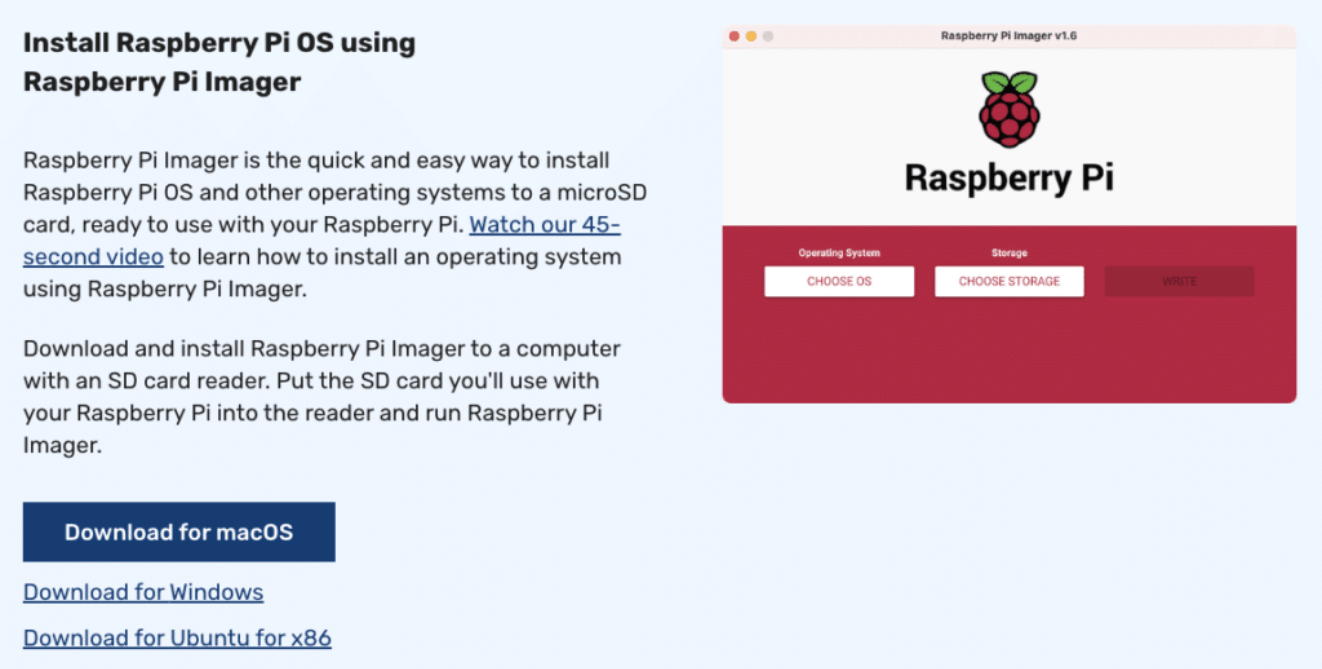
Im Folgenden richtest du deinen Raspberry Pi so ein, dass du ihn bequem von deinem Computer per SSH steuern kannst. Anschließend installierst du dort die Datenbank InfluxDB, in

der deine Messwerte gespeichert werden. Praktischerweise bringt InfluxDB gleich eine Möglichkeit, die Daten ansprechend darzustellen. Doch eins nach dem anderen.

Das Betriebssystem auf dem Raspberry Pi installieren

Zunächst benötigst du ein entsprechend konfiguriertes Betriebssystem. Das lässt du auf eine Micro-SD-Karte schreiben, die du dann in deinen Raspberry Pi steckst.

Besonders einfach ist das mit dem kostenlosen **Rasperry Pi Imager**, den du hier herunterladen kannst. Wähle einfach die Version für dein Betriebssystem und starte den Download.

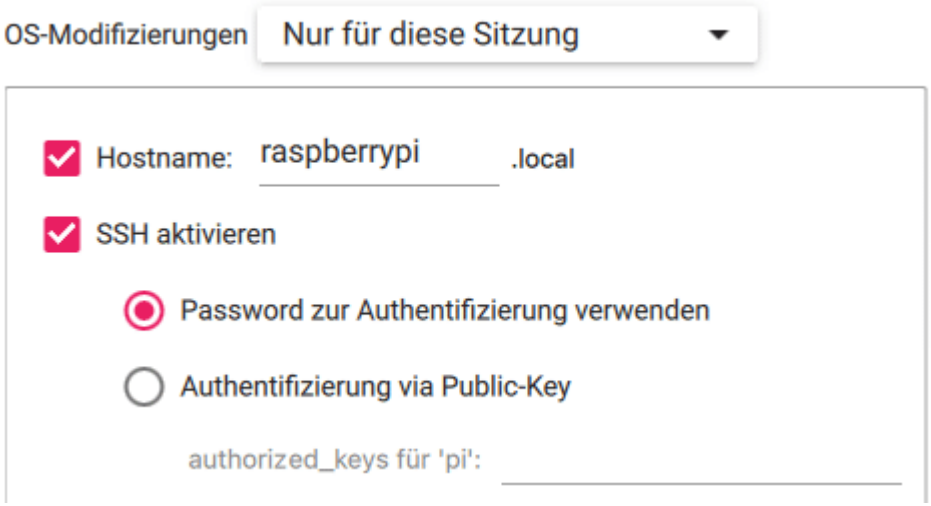


Öffne das Programm nach dem Download und wähle in der Oberfläche unter **Betriebssystem -> Raspberry Pi OS (other) -> Raspberry Pi OS (64-bit)**.

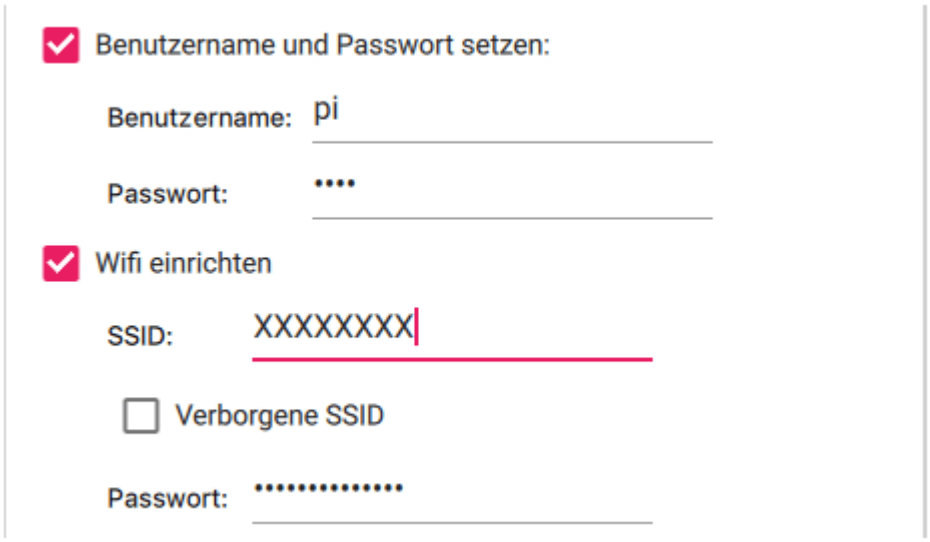
Schließe als nächstes die Micro-SD-Karte, auf die du das Raspberry Pi OS installieren möchtest, an deinen Computer an. Wähle sie anschließend im Feld **SD-Karte** aus.

Bevor du jetzt auf den Button **Schreiben** klickst, wähle zunächst die erweiterten Einstellungen hinter dem Zahnrad-Symbol. Hier kannst du gleich den Hostnamen festlegen, SSH aktivieren und auch deine WLAN-Zugangsdaten hinterlegen. Das bedeutet, dass du deinen Raspberry Pi nicht mehr an einen Monitor anschließen musst, um diese Einstellungen vorzunehmen. Später reicht es, die Micro-SD-Karte und das Stromkabel einzustecken.

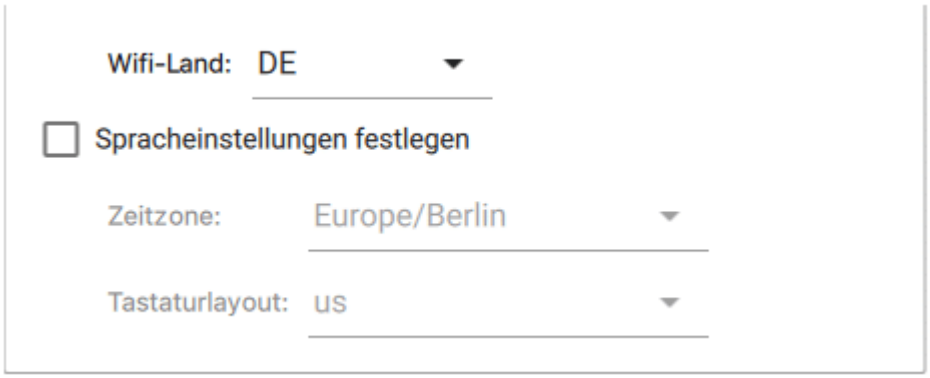
Wähle zunächst im oberen Bereich die folgenden Optionen:



Anschließend legst du deinen Benutzernamen fest und hinterlegst deine WLAN-Zugangsdaten. Die Wahl deines Benutzernamen steht dir natürlich frei – in den folgenden Befehlen verwenden wir hier jedoch das gängige **pi**.



Noch ein Stück weiter unten stellst du noch dein Land und deine Zeitzone ein:



Und das war es. Speichere diese Einstellungen und klicke im Hauptmenü auf **Schreiben**.

Per SSH auf den Raspberry Pi zugreifen

Nachdem das Raspberry Pi OS auf der Micro-SD-Karte und schließlich in deinem Raspberry Pi gelandet ist, starte diesen, indem du das Stromkabel anschließt. Warte nun ein paar Minuten, bis er fertig gebootet hat.

Nun wirst du dich per SSH (Secure Shell) mit dem Raspberry Pi verbinden. SHH ist eine beliebte – und verschlüsselte – Verbindung zwischen zwei Geräten. Hierdurch kannst du auf sichere Art und Weise von deinem Computer auf den Raspberry Pi zugreifen, Software installieren – und später deine Wetterdaten auswerten. So richtest du die Verbindung ein:

MacOS & Linux

Bei diesen beiden Betriebssystemen brauchst du lediglich das Terminal. In Unix-basierten Betriebssystemen ist SSH nämlich schon vorinstalliert. Öffne also das Terminal und tippe den folgenden Befehl ein:

```
sudo ssh pi@raspberrypi.local
```

Hinweis: Falls du einen anderen Benutzer- und Hostnamen vergeben hast, passe den Befehl entsprechend an.

Wenn du nach deinem Passwort gefragt wirst, trage jenes, das du im Raspberry Pi Imager vergeben hast, ein und drücke Enter. Möglicherweise musst du auch noch mal mit einem **yes** bestätigen, dass du die Verbindung aufbauen möchtest. Wenn die Verbindung steht, siehst du die folgende Zeile in deinem Terminal:

Um die Verbindung wieder zu beenden und deinen Raspberry Pi herunterzufahren, trage folgenden Befehl ins Terminal ein:

```
sudo poweroff
```

Windows

In Windows benötigst du eine Software, um dich per SSH zu verbinden – zum Beispiel **PuTTY**. Diese Programm kannst du hier herunterladen.

Installiere PuTTY auf deinem Computer, öffne es und trage die folgenden Daten in den Optionen/Einstellungen ein:

- Host Name: raspberrypi
- Port: 22
- Connection type: SSH

Klicke anschließend auf **Open/Öffnen**. Bei der ersten Verbindung erscheint ein Dialog-Fenster, das dich davor warnt, dass du eine Verbindung zu einem unbekannten Host aufbaust. Diese kannst du mit einem Klick auf **No** schließen.

Logge dich als nächstes mit deinen Zugangsdaten, die du im Raspberry Pi Imager festgelegt hast, ein. Sobald die Verbindung steht, siehst du auch wieder die oben genannte Zeile. Um den Raspberry Pi auszuschalten, verwendest du ebenfalls.

sudo poweroff

INfluxDB 2 auf dem Raspberry Pi installieren

Jetzt wo deine Verbindung steht, kannst du die Datenbank InfluxDB installieren, um die Daten deiner ESP8266 Wetterstation zu speichern. Trage hierfür im Terminal bzw. in PuTTY den folgenden Befehl ein. Kopiere die folgenden Zeilen vollständig, füge sie ins Terminal ein und führe sie mit Enter aus.

```
wget -q https://repos.influxdata.com/influxdata-archive_compat.key
echo '393e8779c89ac8d958f81f942f9ad7fb82a25e133faddaf92e15b16e6ac9ce4c influxdata-archive_compat.key' | sha256sum -c && cat influxdata-archive_compat.key | gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg > /dev/null
echo 'deb [signed-by=/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg] https://repos.influxdata.com/debian stable main' | sudo tee /etc/apt/sources.list.d/influxdata.list

sudo apt-get update && sudo apt-get install influxdb2
```

Nun läuft die Installation im Terminal. Ab und an wirst du um deine Zustimmung zur Installation von Erweiterungspaketen gefragt. Gib diese einfach durch Eingabe von **Y und Enter**.

Nachdem die Installation abgeschlossen ist, erscheint wieder die Zeile **pi@raspberrypi:~$**

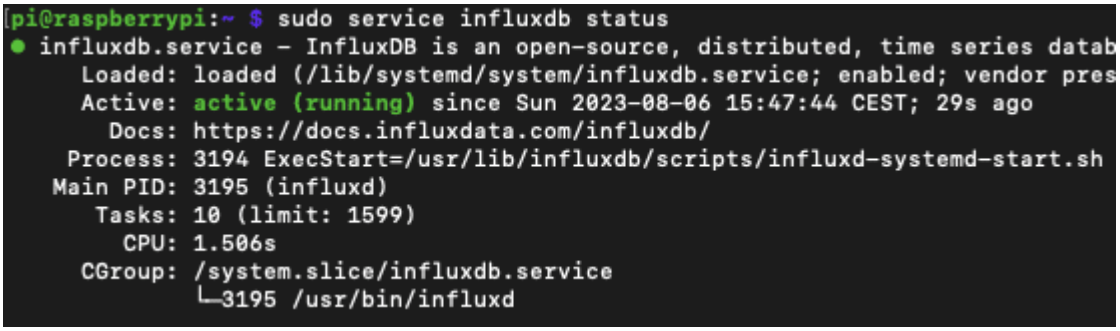
Trage nun folgenden Befehl im Terminal ein, der dafür sorgt, dass InfluxDB beim Start des Raspberry Pi ebenfalls gestartet wird:

```
sudo service influxdb start
```

Anschließend prüfst du noch kurz, ob InfluxDB aktiv ist:

```
sudo service influxdb status
```

Im Terminal erscheint nun einiges an Text – darunter hoffentlich ein grünes **active (running)**:



Drücke **q** auf deiner Tastatur, um wieder zur Kommandozeile zurückzukehren.

Auf die Datenbank zugreifen

Um auf InfluxDB im Browser zugreifen zu können, benötigst du zunächst die IP-Adresse des Raspberry Pi. Diese findest du mit folgendem Befehl im Terminal heraus:

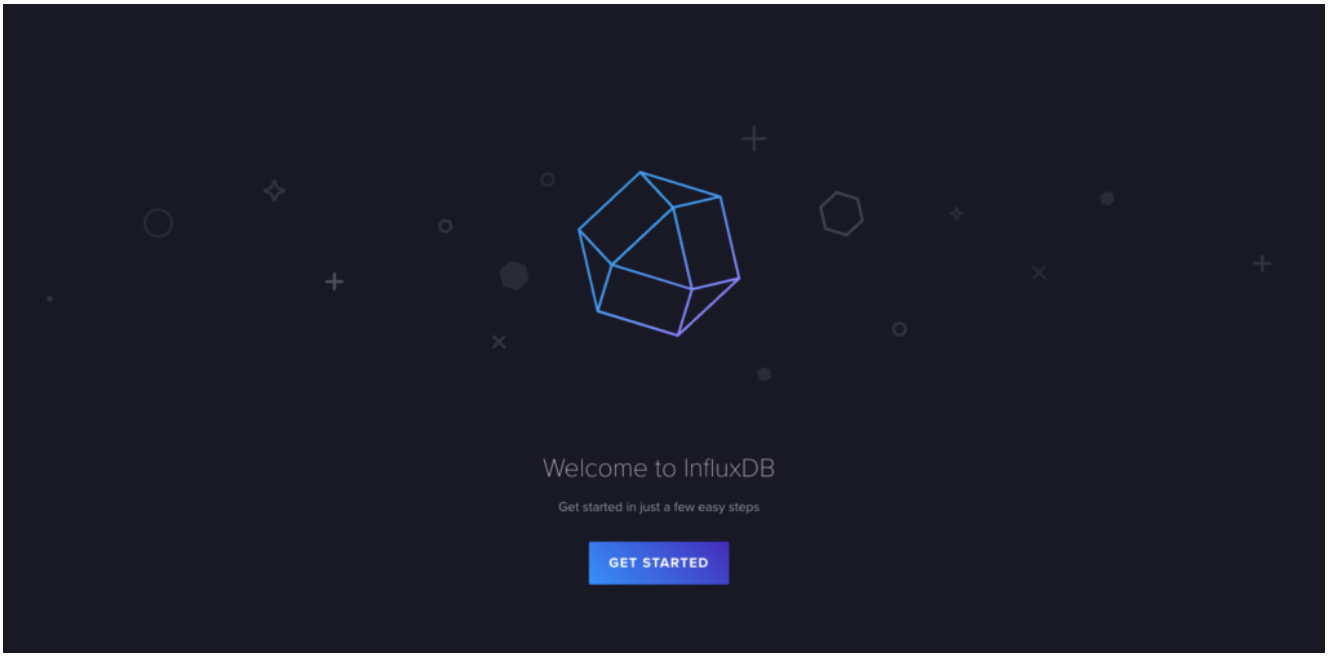
```
hostname -I
```

In der Antwort des Raspberry Pi siehst du ganz vorne die IP-Adresse, in unserem Fall die **192.168.0.129**

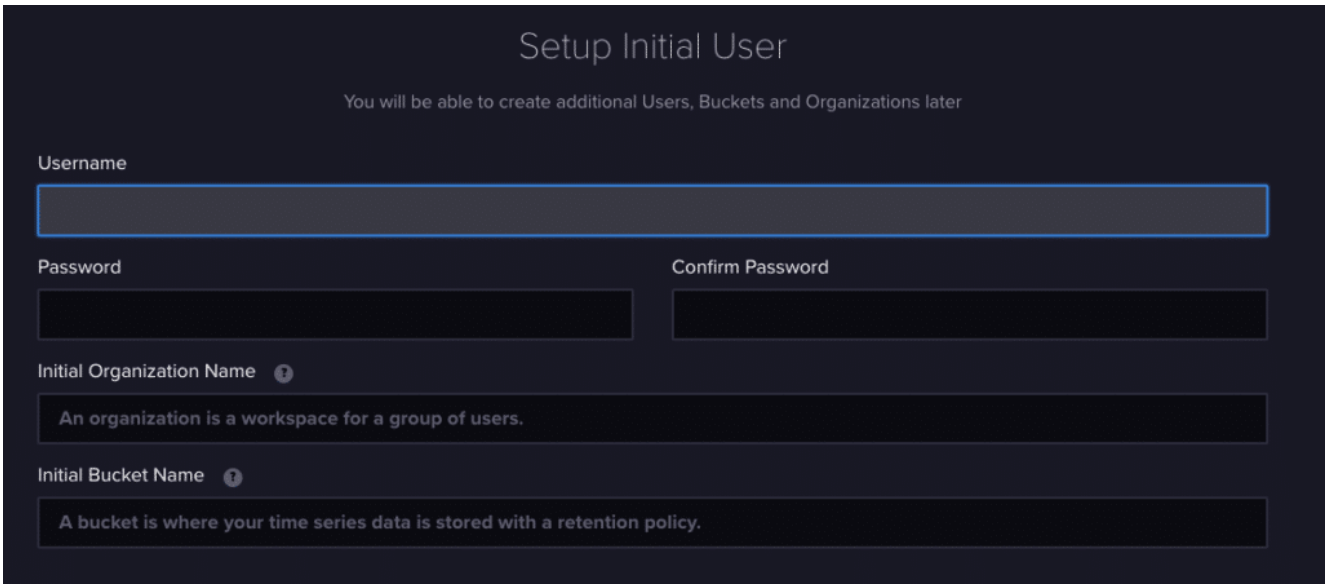
Um nun InfluxDB zu starten, öffne in deinem Browser ein neues Tab oder Fenster und tippe deine IP-Adresse gefolgt vom Port :8086. Die Adresse sieht dann zum Beispiel so aus:

```
192.168.0.129:8086
```

In deinem Browser öffnet sich nun die Startseite von InfluxDB:



Nun kann es losgehen – klicke also auf **Get started**. Es folgt eine Seite, auf der du deine Zugangsdaten festlegst:

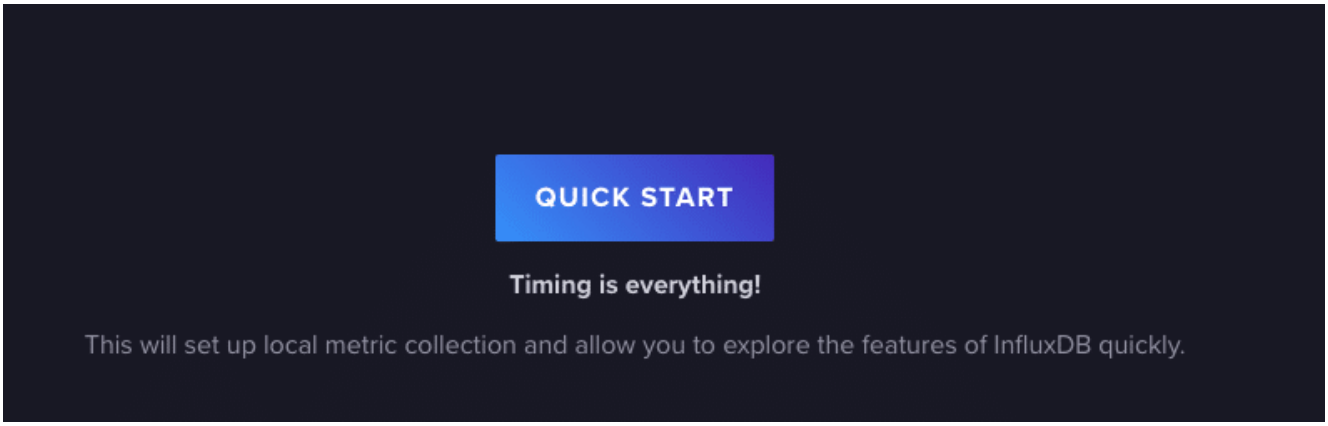


Trage deinen Benutzernamen unter **Username** ein und hinterlege ein sicheres Passwort. Unter **Initial Organization Name** kannst du ebenso deinen Benutzernamen eintragen – solange du nicht wirklich eine Arbeitsgruppe für die Wetterstation hast.

Unter **Initial Bucket Name** kannst du zum Beispiel **ESP8266 Wetterstation** eintragen. Unter diesem Namen findest du später

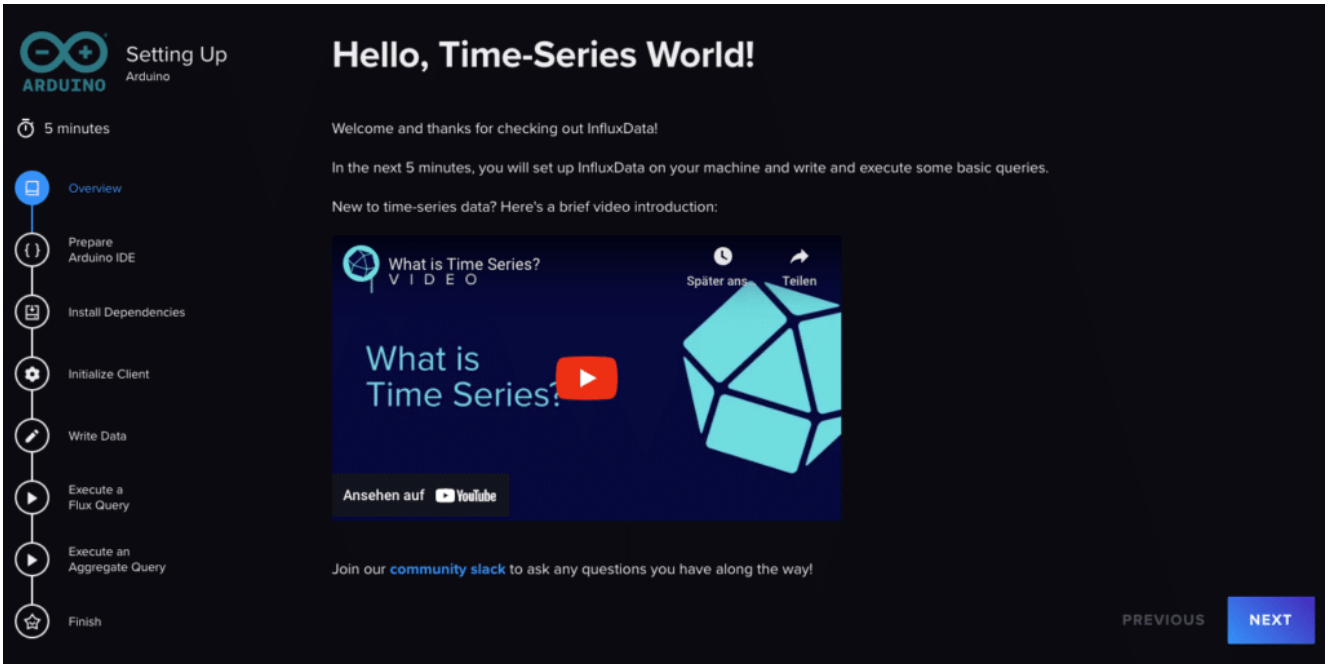
die Datenbank für deine Wetterstation.

Bestätige deine Eingabe und klicke auf der folgenden Seite auf **Quick Start**:



Testweise die WLAN-Signalstärke übertragen

Anschließend findest du auf der Webseite die Kachel **Arduino**. Dahinter verbirgt sich eine Art Tutorial, wie du deinen ESP8266 einrichtest, um Daten an die Datenbank zu übertragen – hier die Stärke deines WLAN-Signals. Dieses Tutorial spielen wir im Folgenden teilweise durch, um einen Sketch zu erhalten, den du danach so erweitern kannst, dass er die Messdaten der Wetterstation überträgt.

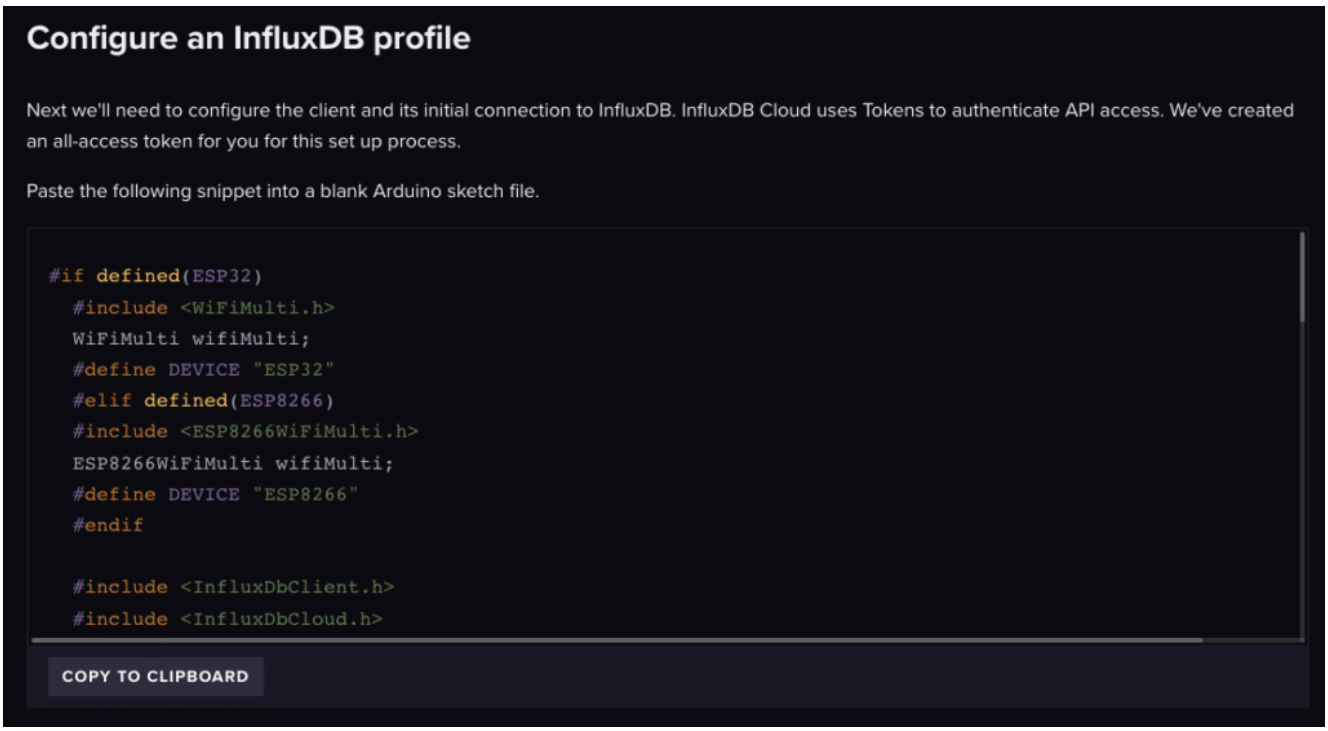


Im Menü links findest du mehrere Punkte – da du den zweiten Punkt **Prepare Arduino IDE** (also den ESP8266 in der IDE verfügbar machen) bereits beim Aufbau der Wetterstation abgeschlossen hast, kannst du ihn gleich überspringen und auf **Install Dependencies** klicken.

Hier erhältst du Informationen zu einer Bibliothek, die du in der Arduino IDE installieren musst, um Daten an InfluxDB übertragen zu können. Öffne also den Bibliotheksmanager und suche nach **InfluxDB**. Im Tutorial heißt die erforderliche Bibliothek **InfluxDB Client for Arduino** – es kann aber gut sein, dass du nur eine Bibliothek namens **ESP8266 Influxdb** findest. Sollte das der Fall sein, installiere einfach diese stattdessen. Wenn du fertig bist, klicke auf **Next**.

Auf der nächsten Seite **Initialize Client** kannst du den Bucket auswählen, in den die Daten übertragen werden sollen:

Gleich darunter findest du Code, den du in einen **leeren Sketch** einfügen sollst. Erstelle also einen neuen Sketch, kopiere den zur Verfügung gestellten Code und füge ihn ein. **Achte unbedingt darauf, dass du die leeren Funktionen Setup und Loop mit diesem Code überschreibst.**



In diesem Beispiel-Sketch musst du noch deine WLAN-Zugangsdaten hinterlegen, damit dein ESP8266 mit dem Raspberry Pi kommunizieren kann:

```
// WiFi AP SSID
#define WIFI_SSID "YOUR_WIFI_SSID"
// WiFi password
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"
```

Trage also deine Daten ein und klicke im Tutorial wieder auf **Next**.

Es folgen wieder zwei Zeilen Code, die du in der Setup-Funktion unterhalb des dort schon vorhandenen Codes einfügen musst:

Der Sketch in diesem Tutorial überträgt die Stärke deines WLAN-Signals an die Datenbank. Die zwei Zeilen fügen den Datenpunkten zwei Tags hinzu – einmal das Gerät und einmal die SSID, also der Name deines WLAN-Netzwerks.

Gleich darunter findest du den Loop des Sketchs. Dieser ist aktuell noch leer – kopiere ihn dir also aus dem Tutorial und überschreibe damit die leere Loop-Funktion in deiner Arduino IDE. Im Loop wird jede Sekunde der **Received Signal Strength Indicator (RSSI, also die Signalstärke des WLAN-Netzes)** gemessen und in der InfluxDB hinterlegt.

Die zwei nächsten Punkte des Tutorials können wir überspringen. Hier geht es um Datenbank-Abfragen, die wir jedoch für unsere Zwecke nicht benötigen.

Der vollständige Beispiel-Sketch

Diese Copy & Paste Arbeit hat etwas Fingerspitzengefühl erfordert. Wenn alles an der richtigen Stelle gelandet ist, sollte dein Sketch wie folgt aussehen:

#if defined(ESP32)

```
#include <WiFiMulti.h>
WiFiMulti wifiMulti;
#define DEVICE "ESP32"
#elif defined(ESP8266)
#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti wifiMulti;
#define DEVICE "ESP8266"
#endif

#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>

// WiFi AP SSID
#define WIFI_SSID "YOUR_WIFI_SSID"
// WiFi password
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"

#define INFLUXDB_URL "DEINE URL (ist vorausgefüllt)"
#define INFLUXDB_TOKEN "DEIN TOKEN (ist vorausgefüllt)"
#define INFLUXDB_ORG "DEINE ORG (ist vorausgefüllt)"
#define INFLUXDB_BUCKET "ESP8266 Wetterstation"

// Time zone info
#define TZ_INFO "UTC2"

// Declare InfluxDB client instance with preconfigured
InfluxCloud certificate
InfluxDBClient   client(INFLUXDB_URL,   INFLUXDB_ORG,
INFLUXDB_BUCKET, INFLUXDB_TOKEN, InfluxDbCloud2CACert);

// Declare Data point
Point sensor("wifi_status");

void setup() {
  Serial.begin(115200);

  // Setup wifi
  WiFi.mode(WIFI_STA);
  wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);

  Serial.print("Connecting to wifi");
```

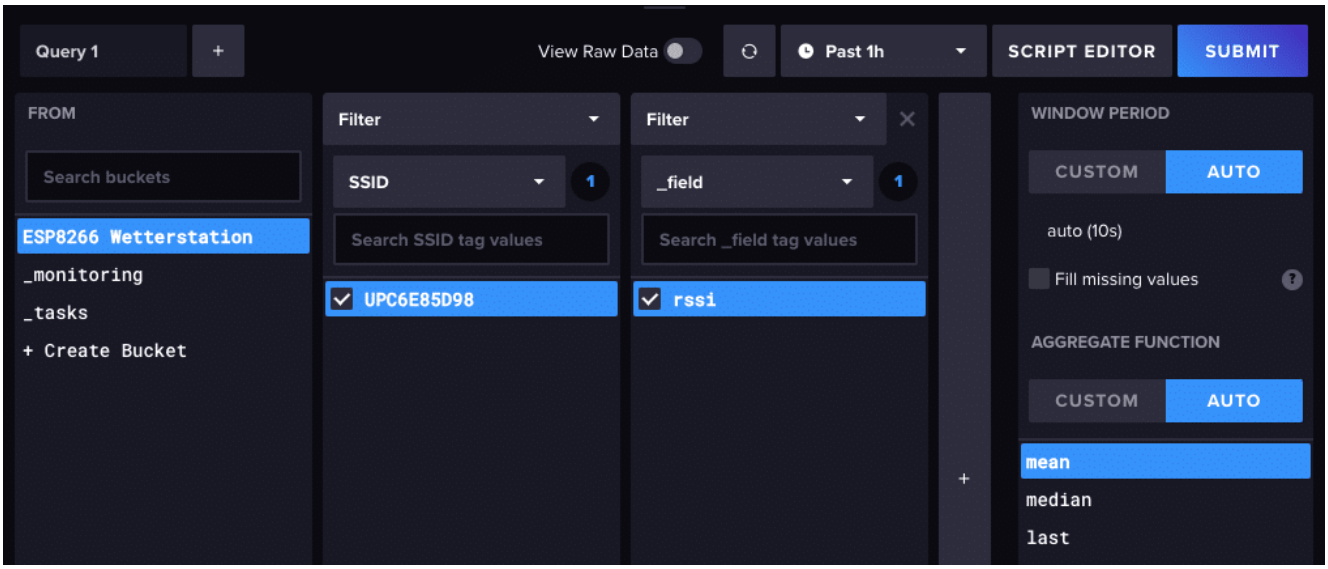
```
  while (wifiMulti.run() != WL_CONNECTED) {
    Serial.print(".");
    delay(100);
  }
  Serial.println();

  // Accurate time is necessary for certificate validation and
writing in batches
   // We use the NTP servers in your area as provided by:
https://www.pool.ntp.org/zone/
  // Syncing progress and the time will be printed to Serial.
  timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");


  // Check server connection
  if (client.validateConnection()) {
    Serial.print("Connected to InfluxDB: ");
    Serial.println(client.getServerUrl());
  } else {
    Serial.print("InfluxDB connection failed: ");
    Serial.println(client.getLastErrorMessage());
  }

  // Add tags to the data point
  sensor.addTag("device", DEVICE);
  sensor.addTag("SSID", WiFi.SSID());
}

void loop() {
    // Clear fields for reusing the point. Tags will remain
the same as set above.
    sensor.clearFields();
    // Store measured value into point
    // Report RSSI of currently connected network
    sensor.addField("rssi", WiFi.RSSI());
    // Print what are we exactly writing
    Serial.print("Writing: ");
    Serial.println(sensor.toLineProtocol());
    // Check WiFi connection and reconnect if needed
    if (wifiMulti.run() != WL_CONNECTED) {
      Serial.println("Wifi connection lost");
```

```
}
// Write point
if (!client.writePoint(sensor)) {
  Serial.print("InfluxDB write failed: ");
  Serial.println(client.getLastErrorMessage());
}
Serial.println("Waiting 1 second");
delay(1000);
}
```

Die Daten visualisieren

Jetzt wird es Zeit, sich die übertragenen Daten einmal anzusehen. Öffne hierfür im Menü links den **Data Explorer** über das Symbol mit dem Koordinatensystem und dem Graphen.

Hier kannst du dir im unteren Bereich eine Abfrage (Query) basteln, die die Signalstärke als Graphen darstellt:

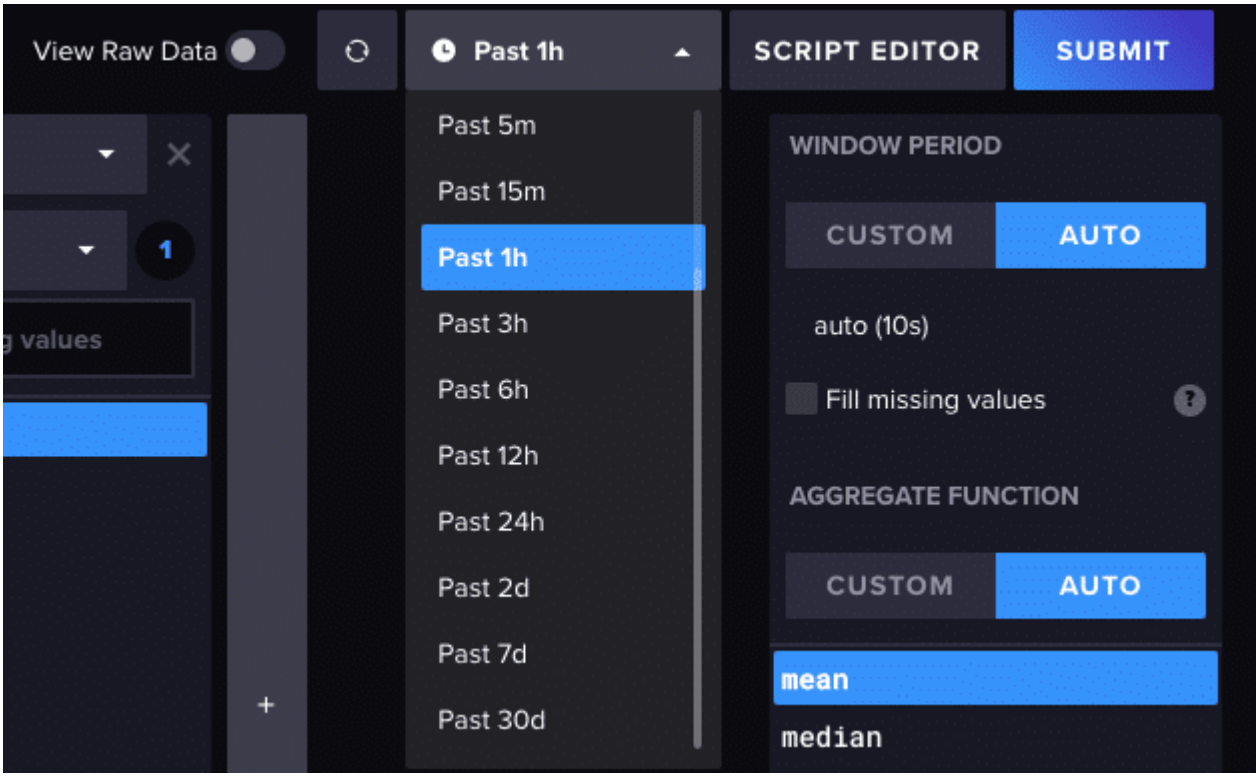


Wähle hierfür links deinen Bucket und im ersten Filter rechts daneben im Dropdown-Menü **SSID** und darunter den Wert, der dem Namen deines WLAN-Netzwerks entspricht. Rechts daneben sollte ein weiterer Filter aufgehen, in dem im Dropdown-Menü bereits **_field** vorausgewählt ist. Als einzigen Wert findest du darin **rssi**, also die Signalstärke. Wähle diesen Wert aus. Falls noch ein weiterer Filter rechts daneben aufgehen, kannst du diesen über das X in der Ecke schließen.

Um diese als Graphen anzuzeigen, musst du nur noch rechts auf **Submit** klicken. Nun sollte im oberen Bereich eine Graph entstehen – hier ein Beispiel, das schon einige Minuten lief:

Deine Abfrage kannst du natürlich zeitlich anpassen.

Voreingestellt ist ein Zeitraum von einer Stunde (Past 1h). Über das entsprechende Dropdown kannst du auch andere Zeiträume einstellen.



Ganz oben findest du auch die Auswahl **Custom Time Range** – hierüber kannst du den Zeitraum noch genauer über einen Kalender einstellen. Sobald du deine Auswahl getroffen hast, klicke links neben dem Dropdown-Menü auf den Refresh-Button oder auf Submit. Über diese beiden Buttons kannst du den Graphen auch aktualisieren, um die aktuellen Daten miteinzubeziehen.

Du überträgst nun bereits Daten von deinem ESP8266 an deine Datenbank auf dem Raspberry Pi – wenn auch „nur“ die Stärke des WLAN-Signals. Nun wird es Zeit, die Messdaten zu übertragen.

Die Daten der ESP8266 WEtterstation

übertragen und anzeigen

Um statt des WLAN-Signals die Messdaten zu übertragen, fügst du den Sketch aus dem Tutorial und deinen bisherigen Sketch der Wetterstation zusammen – und änderst außerdem noch zwei kleinere Stellen im Code.

Kopiere hierfür Stück für Stück aus deinem Wetterstation-Sketch die verschiedenen Teile (Bibliotheken und Definitionen, Setup sowie Loop) und füge sie nacheinander in den Sketch des Tutorials ein. Wenn du das getan hast, sieht dein vollständiger Sketch wie folgt aus:

```
#if defined(ESP32)
#include <WiFiMulti.h>
WiFiMulti wifiMulti;
#define DEVICE "ESP32"
#elif defined(ESP8266)
#include <ESP8266WiFiMulti.h>
ESP8266WiFiMulti wifiMulti;
#define DEVICE "ESP8266"
#endif

#include <InfluxDbClient.h>
#include <InfluxDbCloud.h>

// WiFi AP SSID
#define WIFI_SSID "YOUR_WIFI_SSID"
// WiFi password
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"

#define INFLUXDB_URL "DEINE URL (ist vorausgefüllt)"
#define INFLUXDB_TOKEN "DEIN TOKEN (ist vorausgefüllt)"
#define INFLUXDB_ORG "DEINE ORG (ist vorausgefüllt)"
#define INFLUXDB_BUCKET "ESP8266 Wetterstation"

// Time zone info
#define TZ_INFO "UTC2"

// Declare InfluxDB client instance with preconfigured
```

```
InfluxCloud certificate
InfluxDBClient   client(INFLUXDB_URL,   INFLUXDB_ORG,
INFLUXDB_BUCKET, INFLUXDB_TOKEN, InfluxDbCloud2CACert);

// Declare Data point
Point sensor("wifi_status");

#include "DHT.h"
#include "Wire.h"
#include "Adafruit_BMP085.h"
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define DHTPIN D4
#define DHTTYPE DHT22

Adafruit_BMP085 bmp;

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
-1);

float tempDHT22;
float tempBMP180;
float humidity;
float pressure;

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);

  // Setup wifi
  WiFi.mode(WIFI_STA);
  wifiMulti.addAP(WIFI_SSID, WIFI_PASSWORD);

  Serial.print("Connecting to wifi");
  while (wifiMulti.run() != WL_CONNECTED) {
    Serial.print(".");
    delay(100);
```

```
  }
  Serial.println();

  // Accurate time is necessary for certificate validation and
writing in batches
  // We use the NTP servers in your area as provided by:
https://www.pool.ntp.org/zone/
  // Syncing progress and the time will be printed to Serial.
  timeSync(TZ_INFO, "pool.ntp.org", "time.nis.gov");


  // Check server connection
  if (client.validateConnection()) {
    Serial.print("Connected to InfluxDB: ");
    Serial.println(client.getServerUrl());
  } else {
    Serial.print("InfluxDB connection failed: ");
    Serial.println(client.getLastErrorMessage());
  }

  // Add tags to the data point
  sensor.addTag("device", DEVICE);
  Serial.begin(115200);
  dht.begin();

  if (!bmp.begin()) {
    Serial.println("Sensor BMP180 nicht gefunden!");
    while (1) {}
  }

   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {   //
Display-Addresse: 0x3C für Groesse 128x64px
    Serial.println(F("SSD1306 allocation failed"));
    for (;;)
      ;
  }
  display.setTextSize(1);       //Schriftgröße
  display.setTextColor(WHITE);  //Schriftfarbe


  display.clearDisplay();
```

```
  display.display();
}

void loop() {
  tempDHT22 = dht.readTemperature();
  tempBMP180 = bmp.readTemperature();
  humidity = dht.readHumidity();
  pressure = bmp.readPressure();

  Serial.print("Temperatur DHT22: ");
  Serial.print(tempDHT22);
  Serial.println("*C");

  Serial.print("Temperatur BMP180: ");
  Serial.print(tempBMP180);
  Serial.println("*C");

  Serial.print("Luftfeuchtigkeit DHT22: ");
  Serial.print(humidity);
  Serial.println("%");

  Serial.print("Luftdruck BMP180: ");
  Serial.print(pressure / 100);
  Serial.println("hPa");
  Serial.println();

  display.clearDisplay();
  display.setCursor(10, 10);
  display.println("Temp.:  " + String(tempBMP180) + " *C");
  display.setCursor(10, 29);
  display.println("Luftf.: " + String(humidity) + " %");
  display.setCursor(10, 48);
   display.println("Luftd.: " + String(pressure / 100) + "
hPa");

  display.display();

  // Clear fields for reusing the point. Tags will remain the
same as set above.
  sensor.clearFields();
```

```
  // Store measured value into point
  sensor.addField("temp", tempBMP180);
  sensor.addField("humidity", humidity);
  sensor.addField("pressure", pressure/100);

  // Print what are we exactly writing
  Serial.print("Writing: ");
  Serial.println(sensor.toLineProtocol());

  // Check WiFi connection and reconnect if needed
  if (wifiMulti.run() != WL_CONNECTED) {
    Serial.println("Wifi connection lost");
  }

  // Write point
  if (!client.writePoint(sensor)) {
    Serial.print("InfluxDB write failed: ");
    Serial.println(client.getLastErrorMessage());
  }



  Serial.println("Waiting 1 second");
  delay(1000);
}
```

Im Code oben findest du zwei Änderungen, die wir uns jetzt näher anschauen. Zunächst streichst du den Tag **SSID** in der Setup-Funktion, da wir das nicht mehr brauchen. Übrig bleibt das **device** (also ESP8266) – hierüber können wir später im **Data Explorer** die Messdaten leichter finden.

```
// Add tags to the data point
sensor.addTag("device", DEVICE);
```

Außerdem müssen statt der Signalstärke unsere Messdaten für Temperatur, Luftfeuchtigkeit und -druck übertragen werden. Das geschieht im Loop mit folgendem Code:

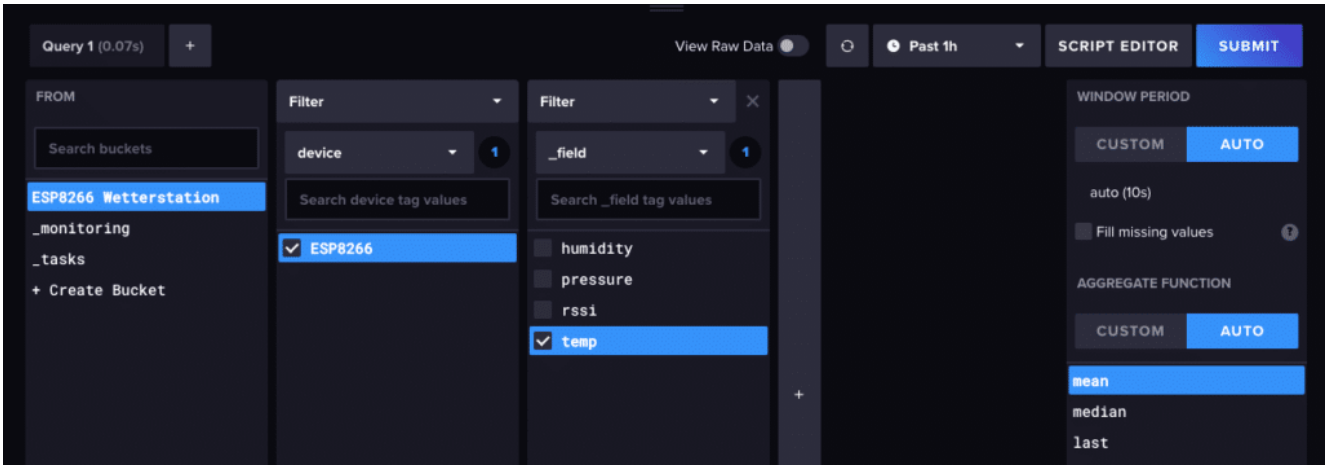
```
  // Store measured value into point
  sensor.addField("temp", tempBMP180);
  sensor.addField("humidity", humidity);
  sensor.addField("pressure", pressure/100);
```

Hiermit werden die drei **Felder** (fields) mit den Namen **temp, humidity und pressure** in der Datenbank angelegt und mit den aktuellen Daten der Sensoren versorgt. Diese stammen aus unseren Variablen **tempBMP180, humidity und pressure**. Wie du siehst teilst du den Wert in der Variablen **pressure** noch durch 100, um statt Pascal (Pa) Hektopascal (hPa) zu erhalten.

Und das war auch schon alles. Achte darauf, dass deine korrekten WLAN-Zugangsdaten im Sketch eingetragen sind und lade ihn auf deinen ESP8266.

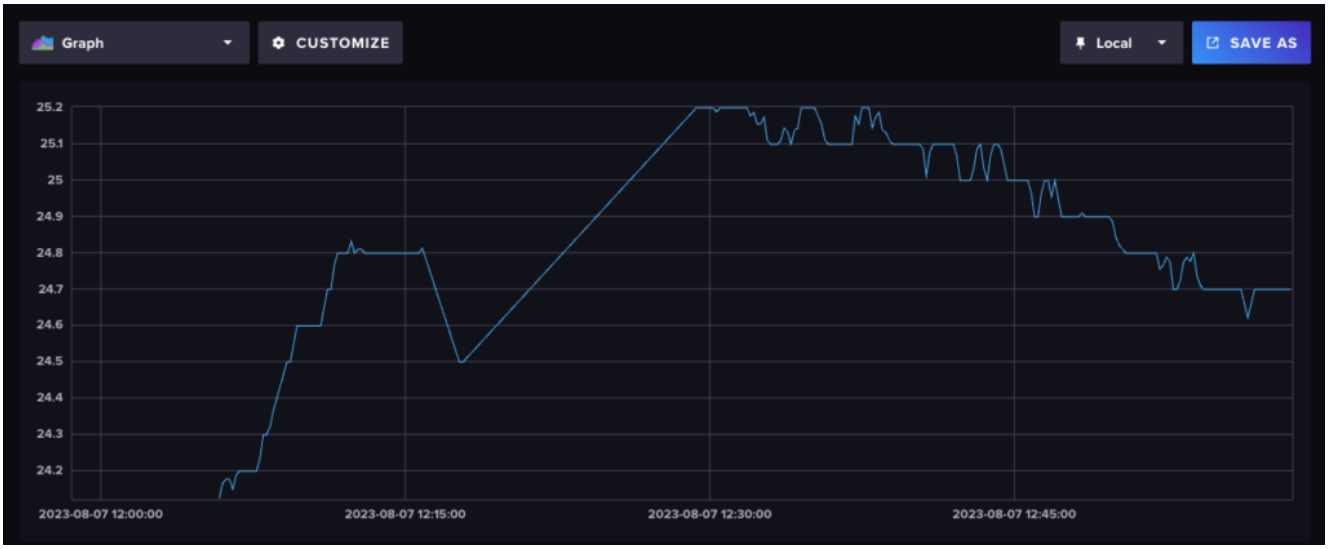
Die Daten im Data Explorer anzeigen

Als nächstes schaust du im Data Explorer nach, ob die Daten auch wie gewünscht deine Datenbank erreichen. Klicke hierfür links auf den entsprechenden Menüpunkt und erstelle deine Abfrage wie folgt:



Hier pickst du dir testweise die Temperatur über die Variable **temp** heraus. Wenn du deine Abfrage erstellt hast, klicke auf den Button **Submit**. Im oberen Bereich der Seite sollte nun ein

Graph zu sehen sein – hier wieder ein Beispiel, das schon einige Zeit lief:



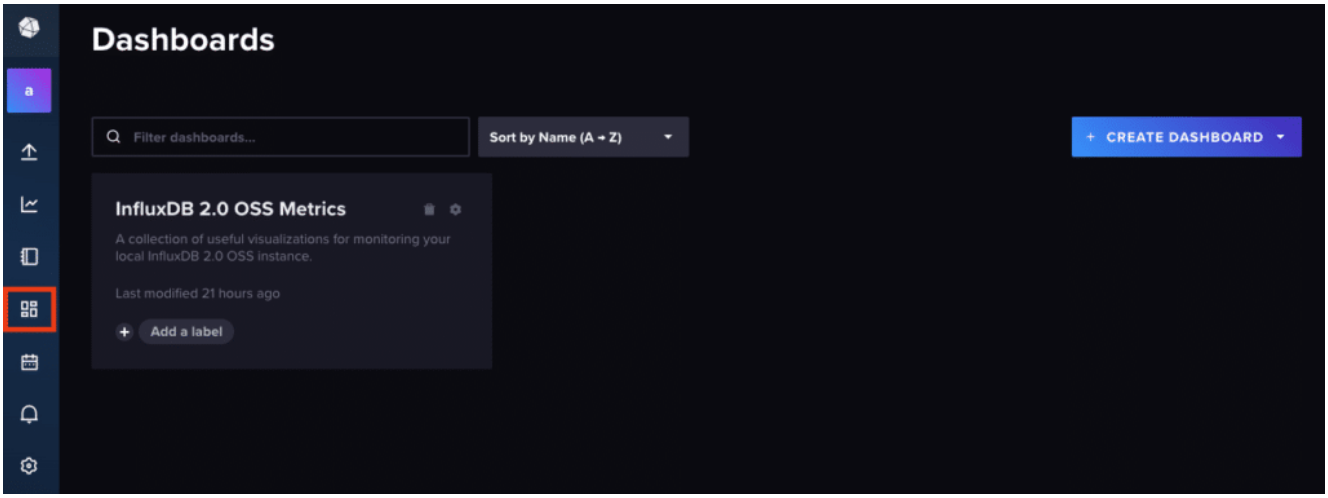
Wie du hier siehst begann die Aufzeichnung kurz nach 12 Uhr am 7. August 2023 mit Werten um die 24 °C. Gegen 12:20 Uhr brach die Verbindung zur Datenbank ab (die Wetterstation war aus) und begann wieder gegen 12:30 Uhr. Diese Zwischenzeit wird im Graphen als Verbindungslinie zwischen dem letzten erhaltenen und ersten wieder verfügbaren Datenpunkt dargestellt. Anschließend sank die Temperatur kontinuierlich von 25 °C auf circa 24,7 °C.

Hinweis: Auf der rechten Seite findest du den Punk **Aggregate Function** – hier kannst du einstellen, ob du von den Temperaturwerten den Durchschnitt (mean), den Median oder immer den letzten aktuellen Wert (last) sehen möchtest. Die letzte Option zeigt dir immer den aktuellen, tatsächlichen Wert an. Die beiden anderen glätten hingegen den Graphen, sodass Temperaturausschläge weniger ins Gewicht fallen.

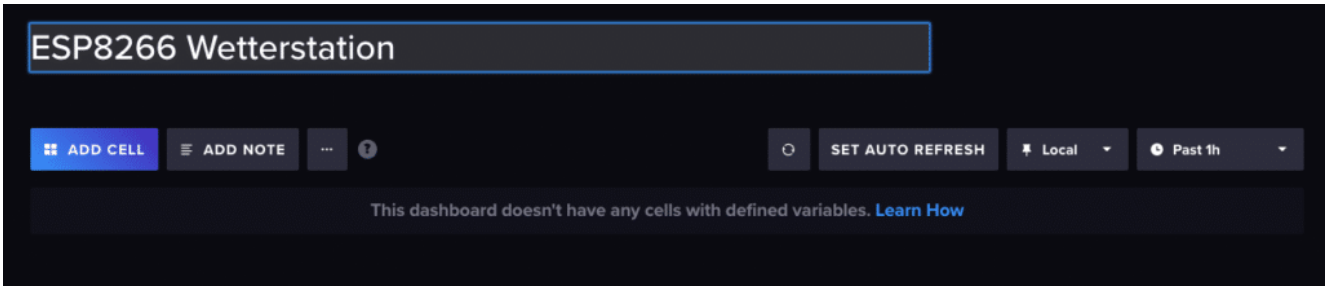
Solange deine ESP8266 Wetterstation also läuft, werden Daten gesammelt. Auf diese kannst du entweder über das Dropdown (z.B. Past 1h) oder spezifisch über den Punkt **Custom Time Range** zugreifen. Es gibt allerdings noch eine weitere spannende Funktion: ein Dashboard.

Die Messdaten auf einem Dashboard anzeigen

Um die aktuelle Abfrage auf einem Dashboard zu speichern, das dir einen schnellen Überblick verschafft, klicke zunächst im Menü links auf den entsprechenden Button:

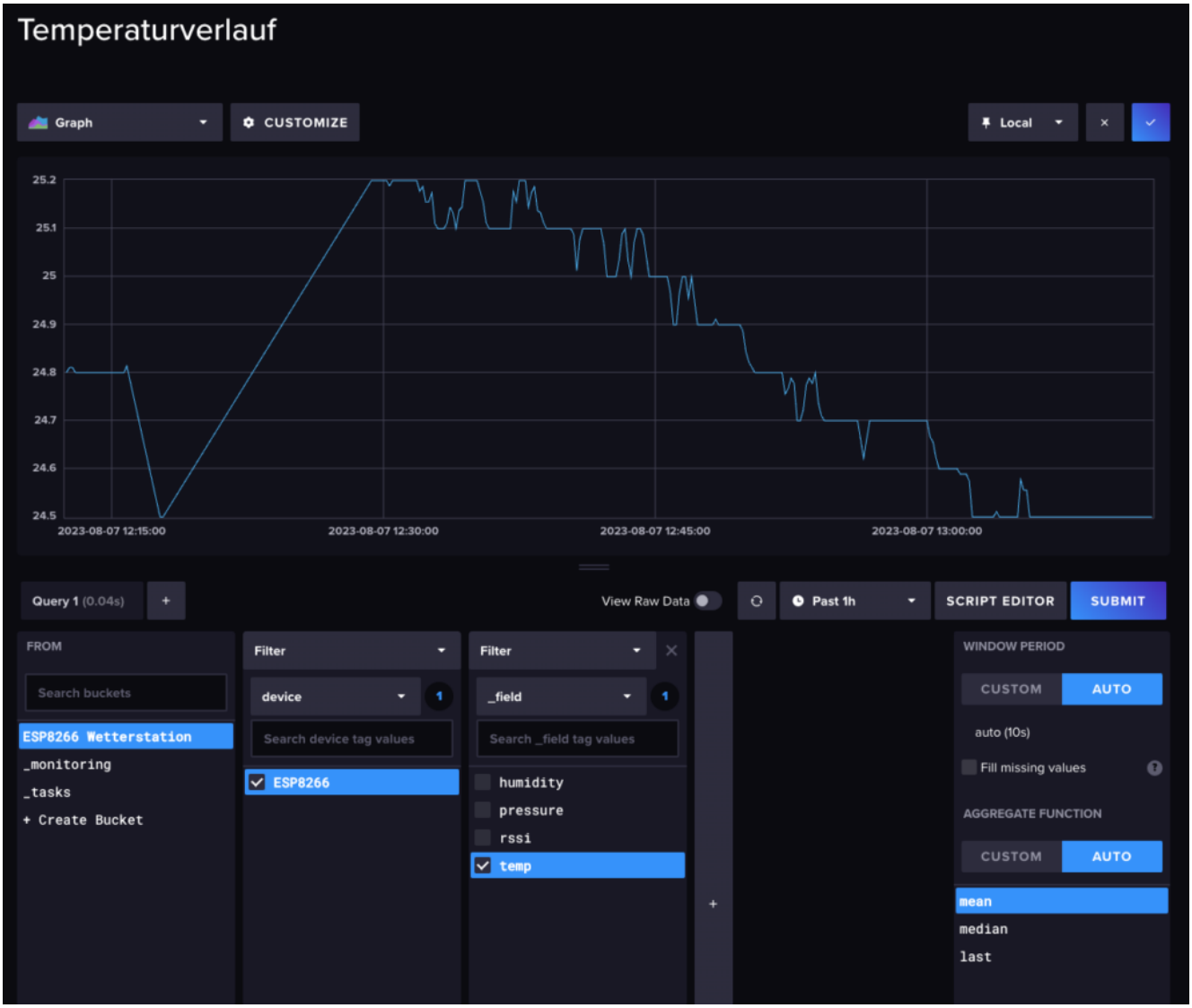


Klicke anschließend rechts auf den Button **Create Dashboard** und anschließend auf **New Dashboard**. Auf der folgenden Seite kannst du oben einen Namen für das Dashboard vergeben, zum Beispiel **ESP8266 Wetterstation:**

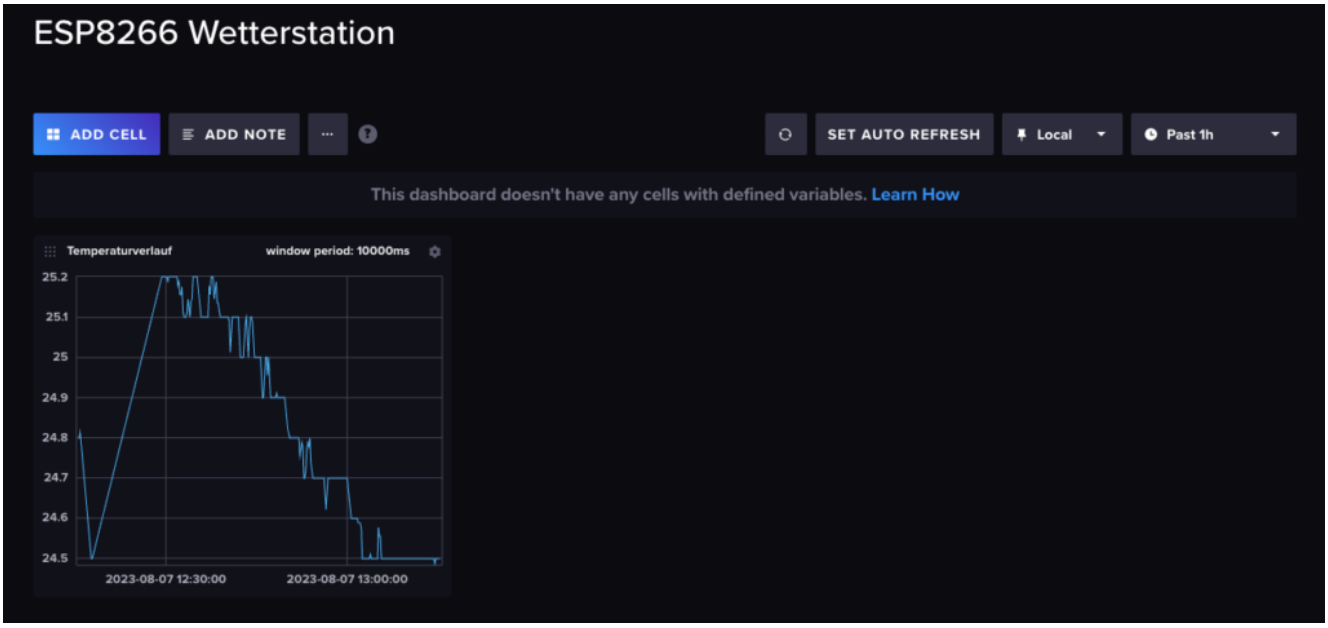


Anschließend kannst du über den Button **Add Cell** eine Kachel hinzufügen – zum Beispiel den gerade gesehenen Temperaturverlauf. Du landest nach deinem Klick wieder in der Ansicht des **Data Explorers**, nur das du diesmal hier die Abfrage für die neue Temperatur-Kachel erstellst.

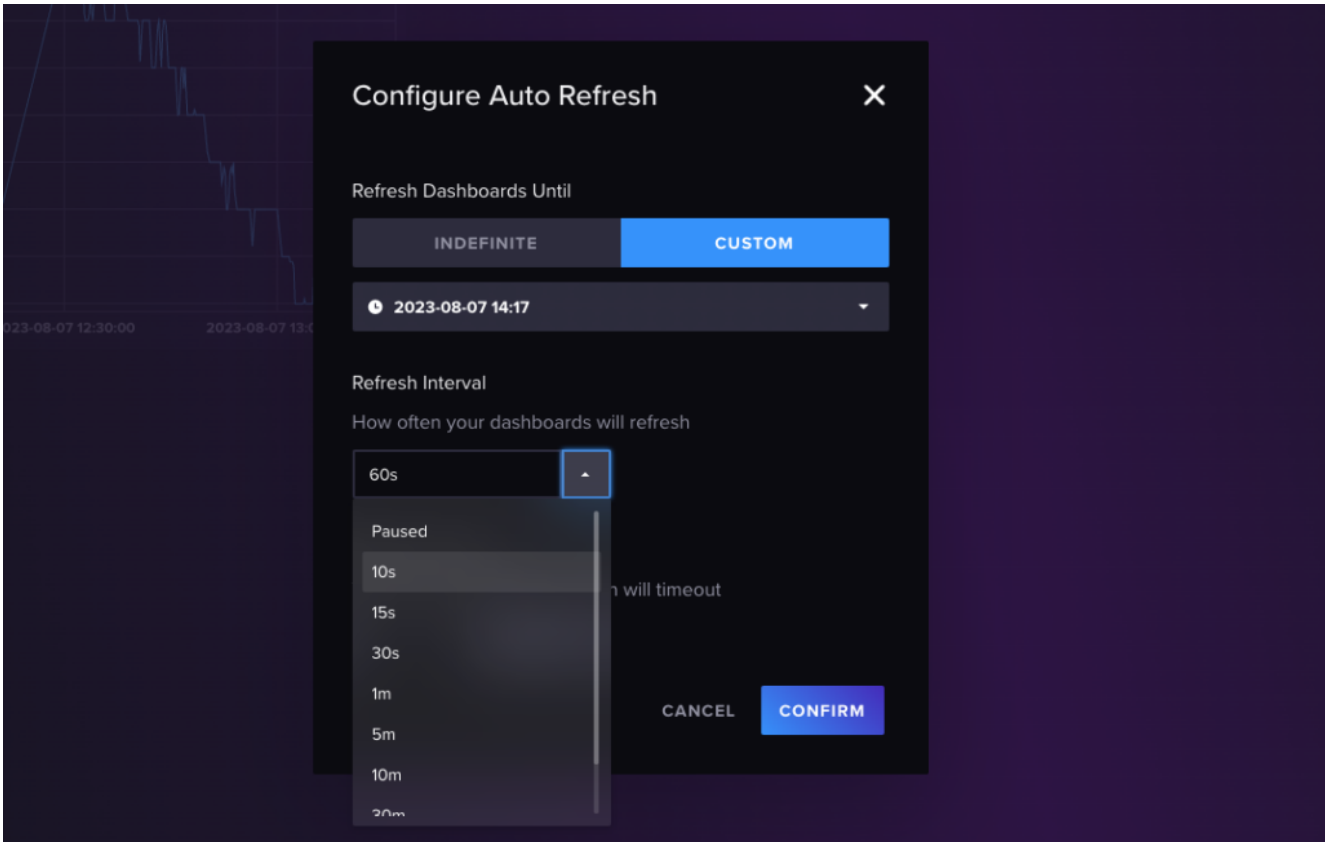
Vergib zunächst wieder ganz oben einen Namen, zum Beispiel **Temperaturverlauf**. Anschließend wählst du im unteren Bereich wieder die Variable **temp** aus.



Alles, was jetzt noch fehlt ist ein Klick auf das Häkchen oben rechts – und schon landest du wieder auf deinem Dashboard, das nun den Temperaturverlauf enthält.



Im oberen Bereich findest du den Button **Set Auto Refresh** – hierüber kannst du einstellen, in welchem Intervall das Dashboard und damit auch der Graph aktualisiert werden soll.



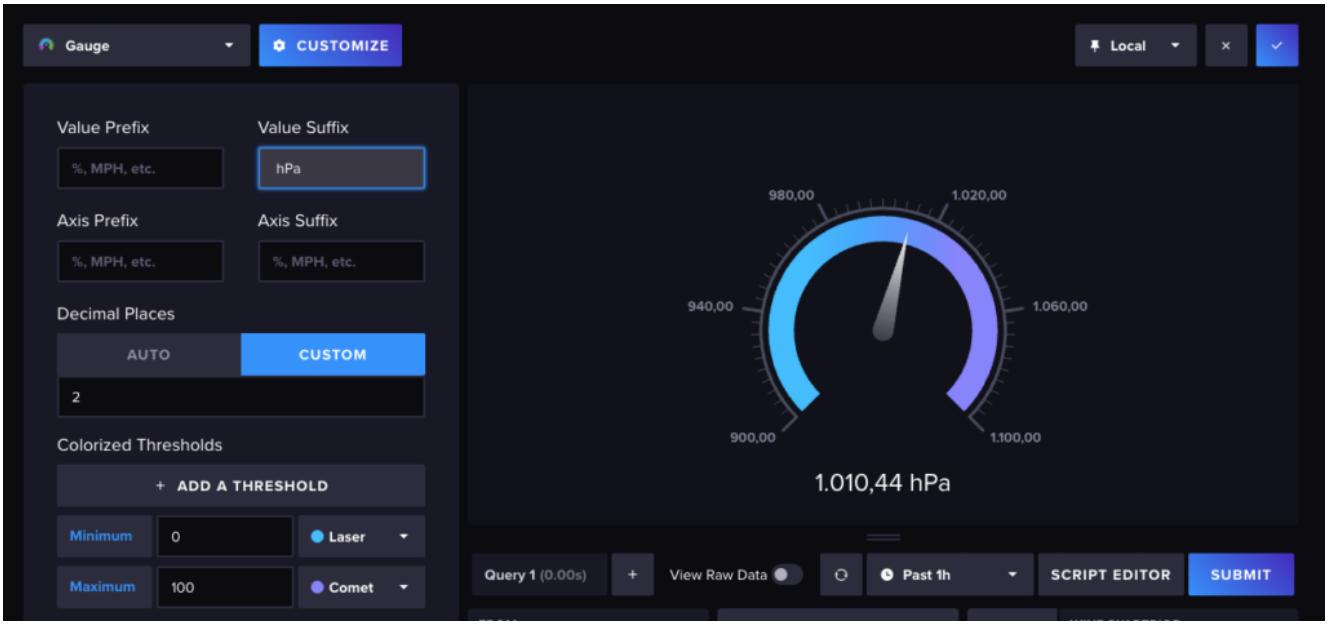
Apropos Graph, du kannst auf dem Dashboard auch die aktuelle Temperatur anzeigen lassen. Klicke hierfür wieder auf **Add Cell**

und erstelle die bekannte Abfrage erneut. Sobald du auf **Submit** geklickt hast, erscheint wieder dein Temperaturverlauf. Oben links findest du ein Dropdown-Menü, in dem du die Anzeige anpassen kannst. Wähle hier den Eintrag **Gauge**.

Nun siehst du nicht mehr den Verlauf, sondern die aktuelle Temperatur:

Vergib dieser Kachel wieder einen Namen und klicke abschließend auf das Häkchen rechts oben. Nun landest du wieder in deinem Dashboard mit den zwei Kacheln zur Temperatur.

Mit deinen beiden anderen Messwerten zu Luftdruck und -feuchtigkeit kannst du genauso verfahren. Noch ein letzter Hinweis zu den Kacheln: Über den Button **Customize** kannst du die Anzeige noch anpassen, was besonders beim Luftdruck sinnvoll ist. Stelle hier unter **Thresholds** einen Bereich von 900 bis 1100 ein, damit der Zeiger auch etwas zum Anzeigen hat. Ebenso kannst du dem Wert noch ein Suffix vergeben – hier also **hPa**.



Warnungen per Nachricht senden

Möchtest du Nachrichten an dich oder jemand anderes senden, wenn ein bestimmter Messwert über- oder unterschritten wird? Im folgenden Tutorial erfährst du, wie du mit dem ESP8266 oder ESP32 Nachrichten per Telegram, WhatsApp und E-Mail verschickst.