Dein eigener ESP8266 Webserver

INHALT

1. VORWORT

Herzlich willkommen! Schön, dass du dich dafür entschieden hast, mit deinem ESP8266 das Tor zum Internet of Things aufzustoßen. Lass uns zunächst einen Blick auf ein paar wichtige Dinge werfen.

1.1 DIESE TEILE BENÖTIGST DU

Um diesem E-Book folgen zu können, benötigst du neben deinem ESP8266 noch ein paar weitere Bauteile:

- Eine rote Standard-LED & Vorwiderstand (z.B. 51Ω) – oder eine andere passende Kombination.
- Einen Temperatursensor: Entweder einen DHT11, DHT22, GY906 oder BMP180. Für den DHT22 benötigst du außerdem einen 10kΩ Widerstand.

Im Verlauf des E-Books erklären wir, wie du diese Sensoren anschließt – und beschäftigen uns vor allem mit dem meist günstigsten der drei, dem BMP180. **Du kannst aber natürlich auch jeden anderen Sensor verwenden – vorausgesetzt du weißt bereits, wie du ihn verwendest.**

Darüber hinaus benötigst du noch ein Breadboard und ein paar Kabel. **Und natürlich ein WLAN-Netz, auf das du zugreifen kannst.** 🙂

1.2 DIE KENNTNISSE SOLLTEST DU SCHON HABEN

Wir erklären zwar ausführlich, wie du deinen ESP8266 Webserver baust und konfigurierst. Trotzdem solltest du bereits wissen,

- wie du die Arduino IDE verwendest und Bibliotheken installierst.
- wie grundlegende Funktionen wie Loops, If-Else in C++ verwendet werden.

Hast du alles? Dann lass uns gleich anfangen!

2. GRUNDLAGEN

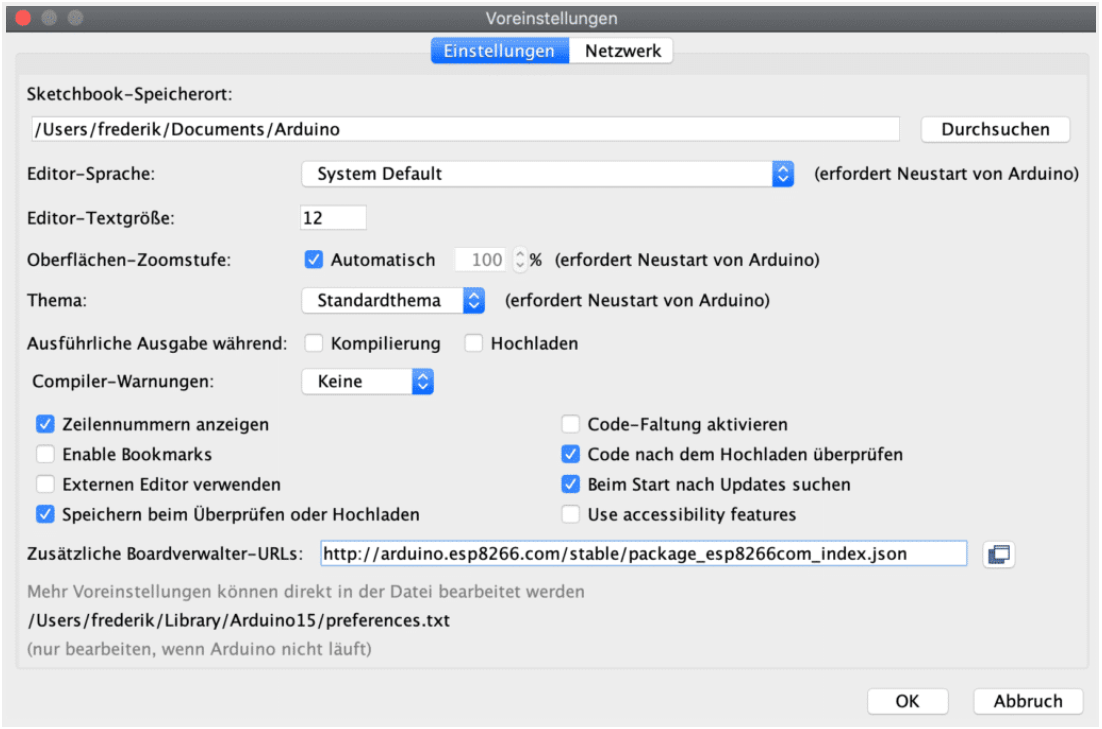
2.1 DEN ESP8266 MIT DER ARDUINO IDE PROGRAMMIEREN

Du kannst deinen ESP8266 genauso programmieren, wie du es von deinem Arduino gewohnt bist. Hierfür musst du allerdings ein paar Vorbereitungen in der Arduino IDE treffen. Das dauert aber nicht länger als 5 Minuten.

Öffne zunächst die Einstellungen der Arduino IDE. Im unteren Bereich findest du das Feld **Zusätzliche Boardverwalter-URLs**.

Trage hier folgende Adresse ein:

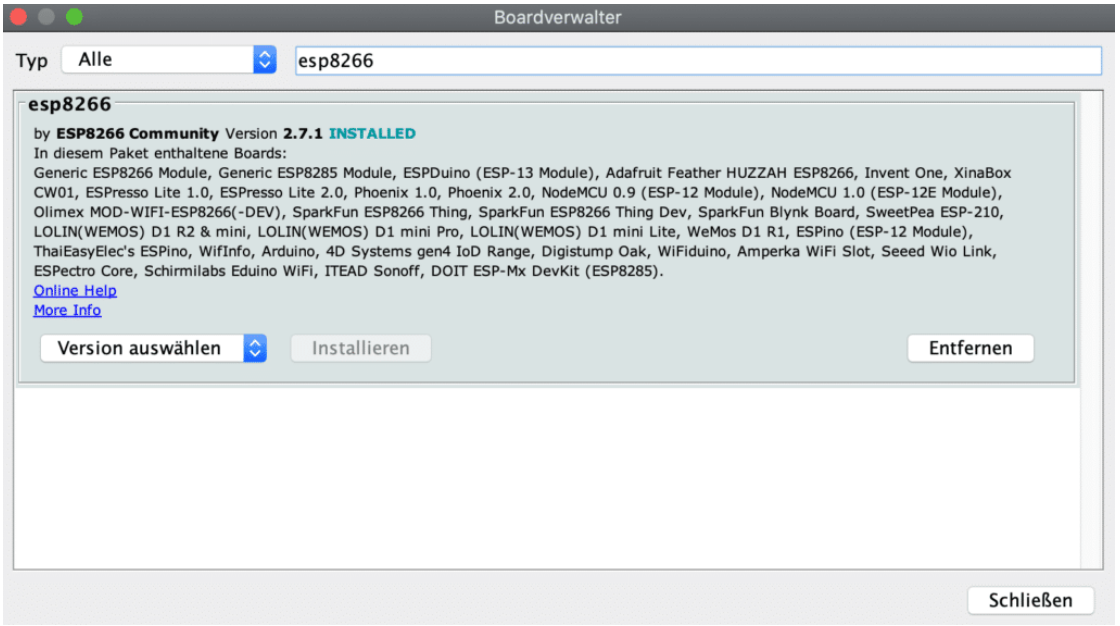
```
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```



Tipp: Wenn du dort später eine andere URL eintragen möchtest – z.B. wenn du einmal mit einem ESP32 experimentierst – schreibe diese einfach mit einem Komma getrennt dahinter. Dann verfügst du in der Arduino IDE über beide Boards.

Schließe nun das Fenster mit einem Klick auf **OK**. Öffne als nächstes das Menü **Werkzeuge** und wähle dort den Menüpunkt **Board** und anschließend **Boardverwalter**.

Suche in dem Fenster, das sich jetzt öffnet, nach **ESP8266**. Scrolle etwas nach unten, bis du den Eintrag **ESP8266 by ESP8266 Community** findest. Noch ein Klick auf **Installieren**, kurz warten und das sollte es gewesen sein.



2.1.2 EIN ERSTER TEST

Lass uns gleich einmal testen, ob die Verbindung zu deinem ESP8266 funktioniert. Erstelle einen neuen Sketch und kopiere folgenden Code hinein:

```
#define LED D0

void setup() {
  pinMode(LED, OUTPUT);
}

void loop() {
  digitalWrite(LED, LOW);
  delay(500);
  digitalWrite(LED, HIGH);
  delay(500);
}
```

Dieser Sketch wird die interne LED deines ESP8266 (am Pin D0) im Halbsekundentakt blinken lassen.

Schließe nun deinen ESP8266 an den USB-Port deines Computers an. Wähle noch einmal im Menü den Punkt **Werkzeuge** und wähle unter **Board** den Eintrag **NodeMCU 1.0** und den Port, an dem dein ESP8266 angeschlossen ist.

Lade jetzt den Sketch hoch. Blinkt die LED?

Hinweis: Solltest du mit dem ESP8266 in der Arduino IDE Probleme beim Hochladen deiner Sketches haben, installiere probeweise eine frühere Version des Boards, oder wähle einen anderen Eintrag im Menü **Board**.

2.2 DEN ESP8266 MIT DEM INTERNET VERBINDEN

Was wäre dein ESP8266 ohne Internetverbindung? Eben.

Lass uns also schnell auf den Code schauen, den du dafür benötigst. **Du kannst das Snippet dann in all deinen weiteren Projekte verwenden.**

2.2.1 DIE PASSENDE BIBLIOTHEK

Für die Internetverbindung benötigst du zunächst die Bibliothek **ESP8266WiFi.h**

Diese sollte in deiner Arduino IDE bereits verfügbar sein – sofern du dort dein Board installiert hast. **Wenn das noch nicht der Fall ist, gehe noch einmal zurück zum entsprechenden Abschnitt.** Du kannst die Bibliothek dann wie folgt am Anfang deines Sketchs – noch vor der Setup-Funktion – einbinden:

```
#include <ESP8266WiFi.h>
```

2.2.2 DEINE ZUGANGSDATEN

Bevor sich dein Controller mit deinem WLAN-Netzwerk verbinden kann, benötigt er die passenden Zugangsdaten. Auch diese legst du am Anfang deines Sketchs zum Beispiel in unveränderlichen Konstanten fest:

```
const char* ssid = "Name deines WLAN-Netzwerks";
const char* password =  "Dein WLAN-Passwort";
```

2.2.3 UND AB INS INTERNET!

Jetzt kann es losgehen. Es gibt prinzipiell mehrere Möglichkeiten, die Verbindung einzurichten und den Verbindungsaufbau im Seriellen Monitor darzustellen. Zentral ist jedoch immer die Funktion **WiFi.begin()** und dass diese **bestenfalls im Setup deines Sketchs ausgeführt wird**, damit für den Loop alles vorbereitet ist.

Trage folgenden Code in deine Setup-Funktion ein:

```
void setup() {

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Ich verbinde mich mit dem Internet...");
  }
  Serial.println("Ich bin mit dem Internet verbunden!");
}
```

Zunächst rufst du die Funktion **WiFi.begin()** auf, der du deine Zugangsdaten als Argumente mitgibst. Der anschließende Loop wird

solange ausgeführt, wie die Verbindung zum Internet noch nicht steht **(WiFi.status() != WL_CONNECTED)** und schreibt jede Sekunde in den Seriellen Monitor, dass die Verbindung aufgebaut wird.

Sobald diese steht, erhältst du die Erfolgsmeldung im Seriellen Monitor. Und das war es auch schon!

Solltest du Probleme beim Verbindungsaufbau haben, prüfe zunächst deine Zugangsdaten genau. Ein weitere Fehlerquelle ist oft die Frequenz des WLAN – dein ESP8266 funktioniert nur in einem Netzwerk mit 2,4 GHz. Solltest du also 5 GHz nutzen, erstelle am besten ein weiteres Netzwerk mit 2,4 GHz. Konsultiere hierfür die Dokumentation deines Providers bzw. Routers.

2.3 EINEN TEMPERATURSENSOR ANSCHLIESSEN

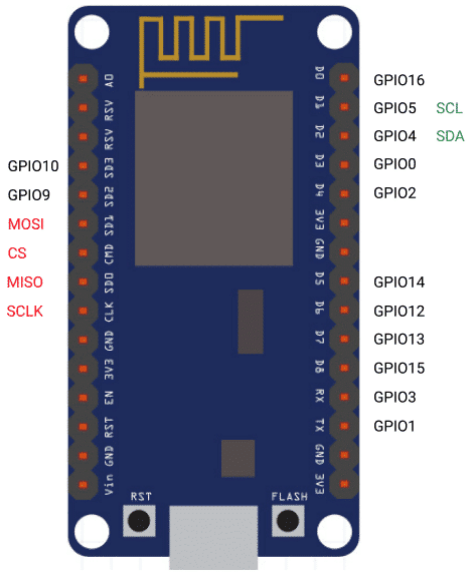
Natürlich gibt es eine Vielzahl von Sensoren, deren Daten du über einen Webserver überprüfen kannst. Wir möchten hier jedoch den wohl beliebtesten Anwendungsfall beschreiben – Temperaturdaten.

Hierfür gibt es auf dem Markt einige günstige Sensoren, die immer wieder zum Einsatz kommen und die sich bewährt haben. Du kannst sie im Handumdrehen anschließen und dank der passenden Bibliotheken ebenso einfach die Temperatur messen.

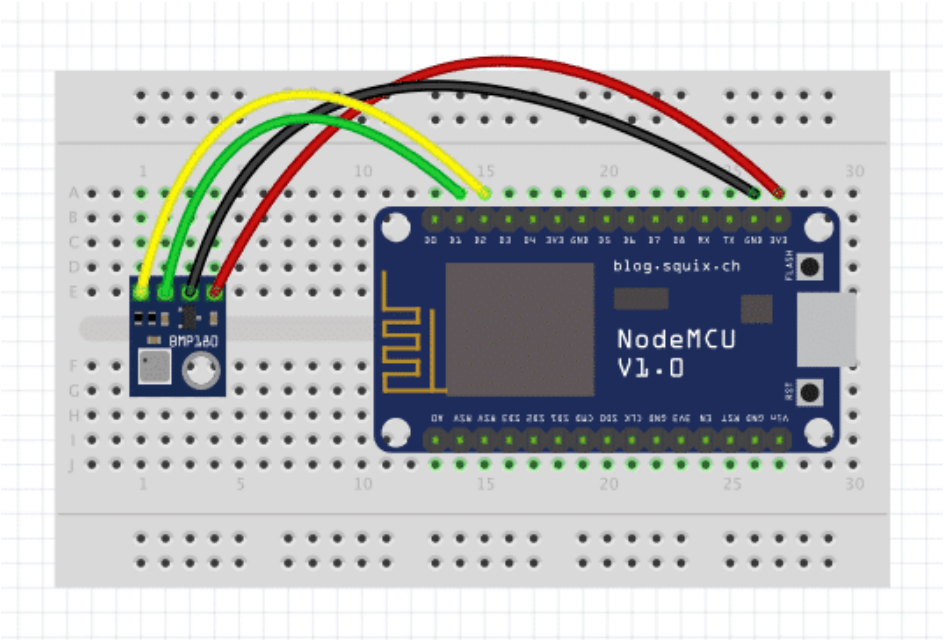
Schauen wir uns drei der bekanntesten Sensoren an: Den BMP180, den GY-906, den DHT22 und DHT11.

2.3.1 DEN BMP180 ANSCHLIESSEN

Diesen Sensor schließt du per I²C an deinem ESP8266 an. Wie du hier siehst, benötigst du hierfür die Pins D1 und D2.



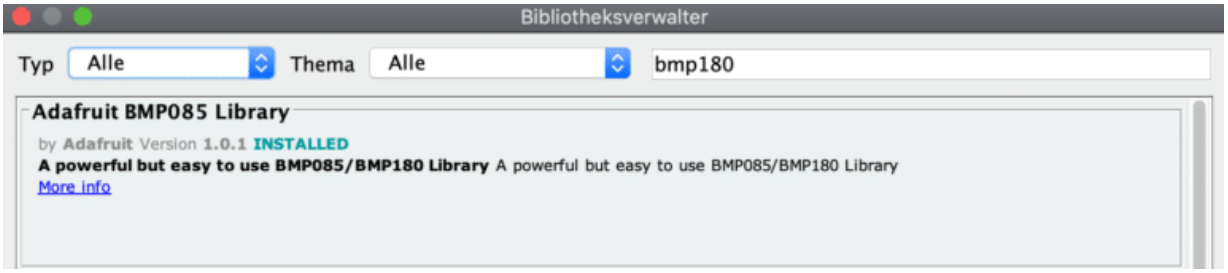
Orientiere dich beim Aufbau an folgender Skizze:



DIE PASSENDE 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.



Installiere nun diese Bibliothek und schließe den Bibltiotheksmanager.

DIE TEMPERATUR MESSEN

Jetzt kann es mit der Messung auch schon losgehen. Kopiere dir den folgenden Sketch und lade ihn auf deinen Arduino hoch:

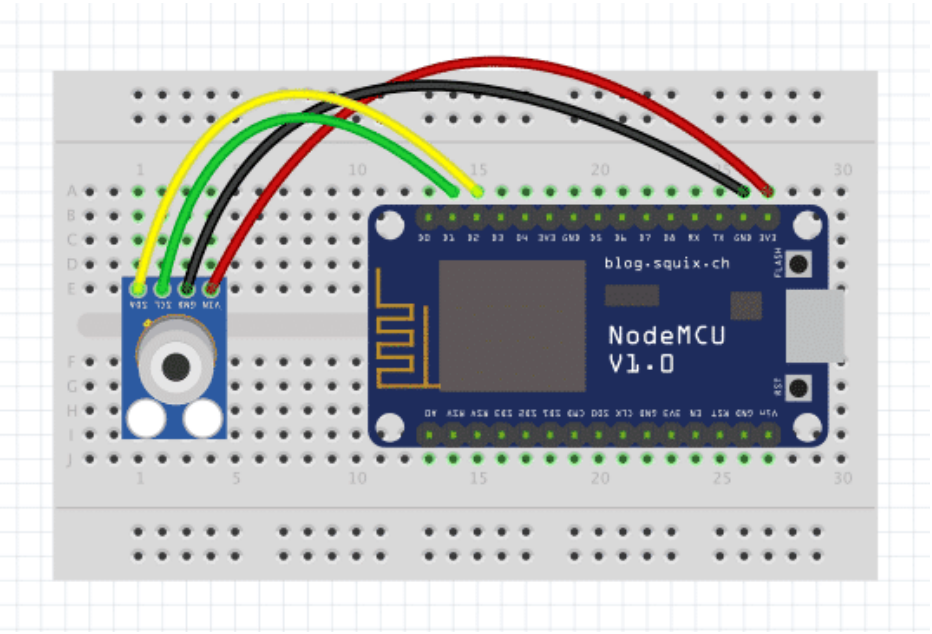
```
#include <Wire.h>
#include <Adafruit_BMP085.h>

Adafruit_BMP085 bmp;

void setup() {
  Serial.begin(115200);
  if (!bmp.begin()) {
      Serial.println("Sensor nicht gefunden!");
      while (1) {}
```

```
  }
}

void loop() {
    Serial.print("Temperatur = ");
    Serial.print(bmp.readTemperature());
    Serial.println(" *C");

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

In deinem Seriellen Monitor sollten jetzt alle 2 Sekunden die Messwerte für die Temperatur in °C erscheinen. Sollte nichts erscheinen, **überprüfe, ob die Baudrate im Sketch und im Seriellen Monitor übereinstimmt oder du SCL und SDA versehentlich verwechselt hast.**

2.3.2 DEN GY-906 ANSCHLIESSEN

Im Prinzip schließt du diesen Sensor genauso an wie den BMP180 – er kommuniziert ebenfalls über I²C. Die Skizze für den Anschluss sieht also ganz ähnlich aus:



DIE PASSENDE BIBLIOTHEK

Auch für den GY-906 gibt es eine Bibliothek, die dir das Leben einfacher macht. Suche im Bibliotheksmanager nach **Adafruit_MLX90614** und installiere die aktuelle Version.

MLX90614 bezieht sich auf den Sensor selbst – GY-906 ist hingegen die Bezeichnung des ganzen Bauteils.

DIE TEMPERATUR MESSEN

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

```
#include <Adafruit_MLX90614.h>
#include <Wire.h>

Adafruit_MLX90614 mlx = Adafruit_MLX90614();

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

void loop() {
  Serial.print("Umgebung = "); Serial.print(mlx.readAmbientTempC());
  Serial.print("*C\tObjekt = "); Serial.print(mlx.readObjectTempC());
  Serial.println("*C");

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

Wie du siehst, bindest du zu Beginn des Sketchs die Bibliotheken für den Sensor und die Kommunikation per I²C ein. Anschließend erstellst du ein Objekt der Bibliothek mit dem Namen **mlx**.