Wetterstation

- Installationsanleitung -

Team Wetterstation

Niklas Schildhauer (ns162)
Carina Szkudlarek (cs324)
Nicolas Wydera (nw073)
Alexander Merker (am206)
Kim Bastiaanse (kb139)

Smart Home Praktikum

CSM SoSe 2021
Hochschule der Medien

Prof. Dr. Gottfried Zimmermann
Tobias Ableitner

1 Einleitung

Das Projekt WeatherIO beschäftigt sich mit der Erstellung einer Wetterstation, die sich einfach an die Bedürfnisse des Nutzers anpassen lässt. Die Wetterstation besteht aus zwei Microcontrollern an denen jeweils Sensoren angeschlossen sind. Ein Raspberry Pi dient als Server für das Speichern von erfassten Messdaten, Kommunikation mit der Datenbank und dem Frontend. Über das Frontend lässt sich die Wetterstation personalisieren und die Wetterdaten in grafischer Form abrufen.

2 Inhalt des Dokuments

In diesem Dokument finden sich sämtliche Informationen, um alle Komponenten in Betrieb zu nehmen sowie benötigte Software zu installieren.

Das Dokument ist in vier Bereiche unterteilt. Der erste Teil beschreibt die allgemeinen Vorbereitungen, welche die Grundlage für die Inbetriebnahme darstellen.
Im Embedded Bereich wird die verwendete Technik, sowie die Verkabelung anhand eines Schaltplans verdeutlicht, notwendige Bibliotheken installiert, sowie der Code für die Microcontroller aufgespielt.
Der Backend Abschnitt beinhaltet alles, was für die Inbetriebnahme des Servers und der Datenbank erforderlich ist.
Im Frontend Bereich wird beschrieben, wie die Web Applikation zu starten ist.

3 Allgemeine Vorbereitung

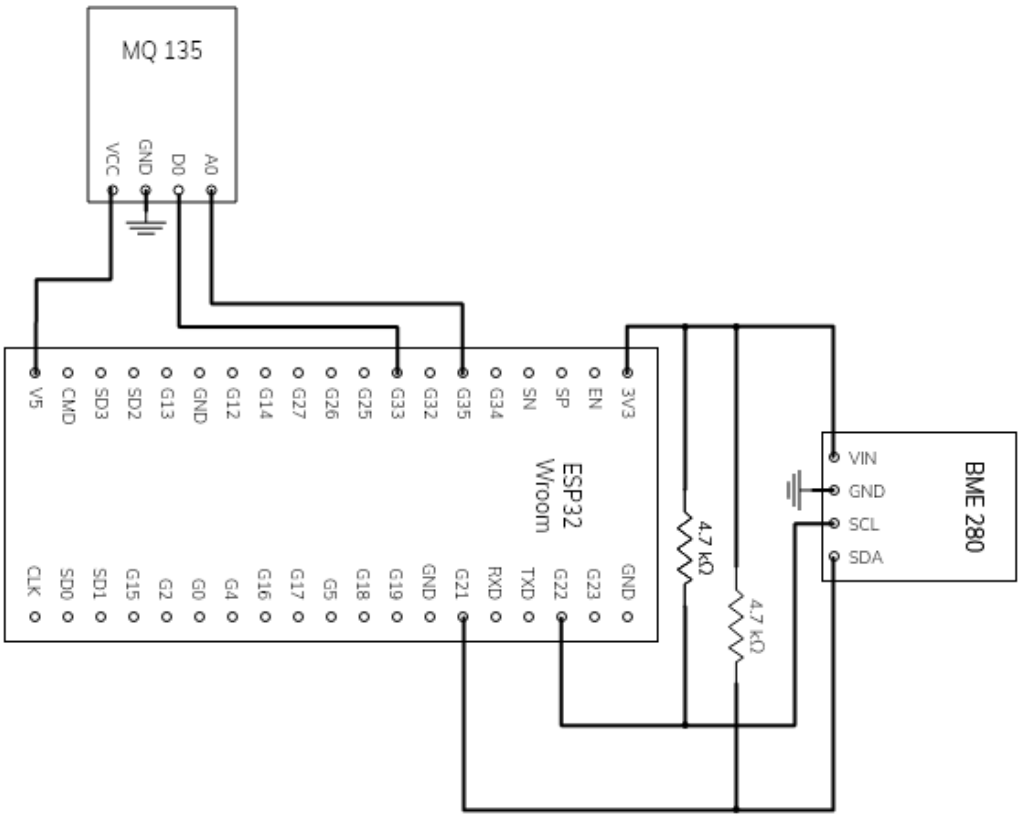
3.1 Zugriff auf das Repository

Damit auf den Code der Wetterstation zugegriffen werden kann muss eine Verbindung zum HdM GitLab Git Repository hergestellt werden. Dazu kann folgender Link genutzt werden https://gitlab.mi.hdm-stuttgart.de/am206/wetterstation.git. Unter Angabe von Benutzername und Passwort kann das Repository per https geklont werden. Alternativ dazu kann auch ein SSH key hinterlegt werden. Eine Anleitung dazu findet sich in GitLab selbst https://gitlab.mi.hdm-stuttgart.de/-/profile/keys.

4 Embedded

4.1 Schaltplan

Wenn die Sensoren und Controller einzeln gekauft werden, gibt der folgende Schaltplan vor, wie die Komponenten angeschlossen werden müssen.



4.1.1 Innenraum

1. Den Luftqualitätssensor (MQ135) anschließen:
    a. Verbinde den Voltage Common Collector (VCC) mit dem 5V-Pin des ESP32.
    b. Verbinde den Ground (GND) mit einem Groundpin am ESP32.
    c. Verbinde den Datenausgang (D0) mit dem Digitalpin G33 am ESP32.
    d. Verbinde den Analogpin (A0) mit dem Analogen Pin G35 am ESP32.

2. Den Temperatur, Luftfeuchtigkeit und Luftdruck Sensor (BME 280) anschließen:
    a. Verbinde den Voltage Common Collector (VCC) mit dem 3V3-Pin des ESP32.
    b. Verbinde den Ground (GND) mit einem Groundpin am ESP32.
    c. Verbinde den Serial Clock Pin (SCL) mit dem G22 Pin am ESP32.
    d. Verbinde den Serial Data Pin (SDA) mit dem G21 Pin am ESP32.
    e. Bringe jeweils einen Pull up Widerstand (4.7kΩ) zwischen VCC & SDA und VCC & SCL an.

4.1.2 Außenraum

1. Den Temperatur, Luftfeuchtigkeit und Luftdruck Sensor anschließen:
    a. Verbinde den Voltage Common Collector (VCC) mit dem 3V3-Pin des ESP32.
    b. Verbinde den Ground (GND) mit einem Groundpin am ESP32.
    c. Verbinde den Serial Clock Pin (SCL) mit dem G22 Pin am ESP32.
    d. Verbinde den Serial Data Pin (SDA) mit dem G21 Pin am ESP32.
    e. Bringe jeweils einen Pull up Widerstand (4.7kΩ) zwischen VCC & SDA und VCC & SCL an.

4.2 Einrichtung der Arduino IDE

Im folgenden Abschnitt wird die Arduino IDE als Toolset für das Hochladen von Konfigurationsdateien und dem Code auf die Microcontroller eingerichtet. Schalte zunächst Zeilennummern im Code ein. Dieser Menüpunkt befindet unter Datei > Voreinstellungen; Checkbox “Zeilennummern”.

4.2.1 Board installieren

Um Code auf das Board ESP32 zu übertragen, muss zunächst das richtige Board heruntergeladen werden.

1. Öffne die Arduino IDE.
2. Öffne Datei > Voreinstellungen.
3. Füge diesen Link unter zusätzliche Boardverwalter-URLs ein: https://dl.espressif.com/dl/package_esp32_index.json
4. Bestätige mit “OK”.
5. Öffne Werkzeuge > Board > Boardverwalter.
6. Gebe im Suchfeld “esp32” ein.

7. Installiere die Version von Espressif Systems.
8. Wähle jetzt in Werkzeuge > Board das Board “ESP32 Dev Module”.

4.2.2 Libraries installieren

1. Öffne Sketch > Bibliothek einbinden > Bibliotheken verwalten
2. Lade die nachfolgenden Libraries herunter
    a. Adafruit BME280 Library (Version 2.1.2) *by Adafruit*
        i. Klicke auf “Alle installieren” um alle Abhängigkeiten automatisch hinzuzufügen
    b. Arduino_JSON (Version 0.1.0) *by Arduino*
    c. WiFi (1.2.7) *by Arduino*
        i. Bei manchen Downloads ist Wifi bereits vorinstalliert, prüfe dies ggf. über den “Installiert” Filter des Bibliotheksverwalters.
    d. AutoConnect (1.2.2) *by Hieromon Ikasamo*
        i. Klicke auf “Alle installieren”
3. Lade die Library für den Raumluftqualitätssensor von Github.
    a. https://github.com/GeorgK/MQ135
    b. Entpacke die Zip-Datei in den “libraries” Ordner deines Arduino IDE Installationsordners.
4. Aktiviere den COM Port unter Werkzeuge/Port. Wähle dabei den verfügbaren COM port aus.
    a. Hinweis: Für Linux/Ubuntu Betriebssysteme muss z.B. der Port /tty/USB0 freigegeben und verwendet werden

4.2.3 Arduino ESP32 Filesystem Uploader installieren

1. Klicke auf den folgenden Link: https://github.com/me-no-dev/arduino-esp32fs-plugin/releases/
2. Lade die ESP32FS-1.0.zip herunter.
3. Suche im Installationsverzeichnis Arduino und öffne den “tools” Ordner.
4. Entpacke die ZIP-Datei im “tools” Ordner.

4.3 In Betrieb nehmen

4.3.1 Innenraum Sensorpaket

Für das Sensorpaket im Innenraum wird die Datei “bme280_mq135” benötigt (wetterstation/embedded/bme280_mq135). Für die Inbetriebnahme muss zunächst die Arduino IDE gestartet werden. Schließe den Innenraumsensor mit einem MicroUSB Kabel an den Computer an.

4.3.1.1 Konfigurationsdatei hochladen

Ändere zunächst die IP-Adresse im Code (Zeile 34) auf die gewünschte IP-Adresse die das Raspberry Pi haben soll (nutze dazu eine statische IP-Adresse). Die Codezeile mit der Adresse sollte in etwa wie folgt aussehen:

```
#define SERVER_IP "192.168.178.190:4201"
```

Die Konfigurationsdatei muss als erstes auf den Flash Speicher des ESP32 übertragen werden. Dazu muss unter Werkzeuge auf “ESP32 Sketch Data Upload” geklickt werden. Im Falle, dass dies nicht auf Anhieb funktioniert, muss der “BOOT” Knopf des ESP32 beim Hochladen permanent gedrückt werden. Dabei wird die Konfigurationsdatei die im Unterverzeichnis “data” liegt, auf den Flash Speicher geschrieben.

4.3.1.2 Code hochladen

Um den Code hochzuladen, klicke auf “Hochladen” (Pfeil links oben | Sketch > Hochladen). Der Code wird automatisch kompiliert und auf das Board übertragen. Im Falle, dass dies nicht auf Anhieb funktioniert, muss auch hier der “BOOT” Knopf des ESP32 beim Hochladen permanent gedrückt werden.

4.3.2 Außenbereich Sensorpaket

Für das Sensorpaket im Außenbereich wird die Datei “bme280” benötigt (wetterstation/embedded/bme280). Für die Inbetriebnahme muss zunächst die Arduino IDE gestartet werden. Schließe den Außenraumsensor mit einem MicroUSB Kabel an den Computer an.

4.3.2.1 Konfigurationsdatei hochladen

Die Konfigurationsdatei muss als erstes auf den Flash Speicher des ESP32 übertragen werden. Dazu muss unter Werkzeuge auf “ESP32 Sketch Data Upload” geklickt werden. Im

Falle, dass dies nicht auf Anhieb funktioniert, muss der “BOOT” Knopf des ESP32 beim Hochladen permanent gedrückt werden. Dabei wird die Konfigurationsdatei die im Unterverzeichnis “data” liegt, auf den Flash Speicher geschrieben.

4.3.2.2 Code hochladen

Ändere zunächst die IP-Adresse im Code (Zeile 34) auf die gewünschte IP-Adresse die das Raspberry Pi haben soll (nutze dazu eine statische IP-Adresse). Die Codezeile mit der Adresse sollte in etwa wie folgt aussehen:
`#define SERVER_IP "192.168.178.190:4201"`

Um den Code hochzuladen, klicke auf “Hochladen” (Pfeil links oben | Sketch > Hochladen). Der Code wird automatisch kompiliert und auf das Board übertragen. Im Falle, dass dies nicht auf Anhieb funktioniert, muss der “BOOT” Knopf des ESP32 beim Hochladen auch hier permanent gedrückt werden.

5 Backend

5.1 Installation

Der Server sowie das Frontend werden auf einem Raspberry Pi gehostet, deshalb müssen auf diesem alle nötigen Programme vorhanden sein. Bitte installiere die folgenden Programme über das Terminal des Raspberry Pi.

5.1.1 NodeJS

Installieren von NodeJS mit dem nachfolgenden Command:

```
sudo curl -sSL https://deb.nodesource.com/setup_14.x | sudo bash -

sudo apt-get install -y nodejs
```

Überprüfen der aktuellen Version mit:

```
node -v
```

5.1.2 npm

Überprüfen der aktuellen Version mit:

```
npm -v
```

5.1.3 Nodemon

Installieren von nodemon als globales npm package

```
sudo npm i -g nodemon
```

Überprüfen der aktuellen Version mit:

```
nodemon -v
```

5.1.3 Ts-node

Installieren von ts-node als globales npm package

```
sudo npm i -g ts-node
```

Überprüfen der aktuellen Version mit:

```
ts-node -v
```

5.1.4 npm-run-all

Installieren von npm-run-all, damit die Microservices parallel über einen Befehl gestartet werden können.

```
sudo npm i -g npm-run-all
```

5.2 Code auf Raspberry Pi auschecken

Wenn nicht bereits vorhanden, muss der Code aus dem HdM GitLab Repository ausgecheckt werden. Dieses ist unter https://gitlab.mi.hdm-stuttgart.de/am206/wetterstation.git zu finden.

5.3 Starten des Servers

Das Backend/der Server können entweder in einem einzelnen Prozess gestartet werden oder zum Debugging bzw. zur Weiterentwicklung bietet es sich an, die npm scripts einzeln in separaten Konsolen zum besseren Überblick bzw. Lesbarkeit zu starten. Für den Normalbetrieb ist allerdings der in 5.2.1 beschriebene Befehl ausreichend.

5.3.1 Normalbetrieb

Starte ein Terminal im Ordner “wetterstation” für:

```
raspi@raspi: ~/wetterstation$ npm run server
```

Hinweis: Bei der Erstinstallation kann dieser Vorgang mehrere Minuten dauern.

Überprüfe gegebenenfalls ob alle Teile des Servers laufen. Benutze dazu den nachfolgenden Befehl. Es müssen 5 Einträge erscheinen.

```
raspi@raspi: ~/wetterstation$ netstat -an | grep 420
```

5.3.2 Betrieb für Debugging/Entwicklung

Starte ein Terminal für den Datenbank-Service:

```
raspi@raspi: ~/wetterstation$ npm run database
```

Starte ein weiteres Terminal für den Authentifizierungsservice:

```
raspi@raspi: ~/wetterstation$ npm run auth
```

Starte ein weiteres Terminal für den Wetterdaten Service zum Empfangen und Ausgeben von Wetterdaten sowie zur Berechnung der Vorhersage:

```
raspi@raspi: ~/wetterstation$ npm run weather-data
```

Starte ein weiteres Terminal für den API-Gateway zur Kommunikation mit dem Frontend:

```
raspi@raspi: ~/wetterstation$ npm run backend
```

Starte ein weiteres Terminal für den Service zum Laden von Personalisierungs Einstellungen aus der DB und OpenAPE:

```
raspi@raspi: ~/wetterstation$ npm run pers
```

6 Frontend

Genau wie der Server muss auch das Frontend auf dem Raspberry Pi gehostet werden. Im Folgenden wird erklärt, wie das Frontend gehostet werden kann und was notwendig ist, um weiter daran zu entwickeln.

6.1 Installation

Das Frontend wurde mit Angular entwickelt, weshalb die Angular CLI global installiert sein muss. Außerdem muss NodeJS installiert sein (siehe 5.1.1)

6.1.1. Angular

Installieren von Angular als globales npm package

```
sudo npm i -g @angular/cli
```

Überprüfen der aktuellen Version mit:

```
ng version
```

Die Wetterstation wurde mit der Version Angular CLI: 11.2.8 entwickelt.

6.1.2. Angular-http-server

Damit das Frontend im gesamten Heimnetzwerk und auch auf Mobilgeräten verfügbar ist, muss ein http server installiert sein. In dieser Anleitung wird beispielhaft der angular-http-server verwendet, es können auch andere (z.B. nginx / Apache) genutzt werden. Durch folgenden Befehl wird der angular-http-server als globales npm package auf dem Raspberry Pi installiert.

```
sudo npm i -g angular-http-server
```

6.2 Lokal starten

Um die Angular Anwendung lokal zu starten ist lediglich der folgende Befehl in einem separaten Terminal auszuführen.

```
raspi@raspi: ~/wetterstation$ npm run frontend-test
```

Durch diesen Befehl wird zuerst npm install ausgeführt, um alle Abhängigkeiten zu installieren und anschließend ng serve --open, sodass der Entwicklungsserver gestartet wird und das Frontend unter der Adresse: http://localhost:4200 erreichbar ist.

Durch das lokale Starten wird die frontend/src/environments/environment.ts Datei als Environment eingelesen, wodurch initial Testdaten verwendet werden. Sollte stattdessen eine Verbindung zum Backend hergestellt werden, dann muss die Variable testData (in src/environments/environment.ts) auf false gesetzt werden:

```
export const environment = {
  production: false,
  testData: true,
  baseURL: '',
};
```

6.3 Im WLAN-Netzwerk hosten

Um die Anwendung auf einem Raspberry Pi zu hosten, muss zuerst die Web-Anwendung für eine produktive Umgebung kompiliert werden. Die Output-Dateien können anschließend mithilfe eines einfachen Http-Servers (z.B. Angular-http-server) gehostet werden.

Wichtig: Aufgrund von CORS muss das Backend die IP-Adresse des Raspberry Pis explizit erlauben. Hierfür muss in der `~/wetterstation/server.js` Datei in der Zeile 28 die IP-Adresse und Port des Raspberry Pis eingetragen werden:

```
26 //Allow CORS, because Angular is running on a different port (4200)
27 const corsOptions = {
28   origin: "http://192.168.178.190:8080",
29   allowedHeaders: [
30     "Origin",
31     "X-Requested-With",
32     "Content-Type",
33     "Accept",
34     "Authorization",
35     "x-xsrf-token"
36   ]
37 };
```

Wichtig: In der environment.prod.ts File muss außerdem ebenfalls die statische IP des Raspberry Pis eingetragen werden. Öffne hierfür unter wetterstation/frontend/src/environments die Datei und ändere die baseURL zu der statischen IP des verwendeten Raspberry Pis.

```
export const environment = {
    production: true,
    testData: false,
    baseURL: 'http://192.168.178.190:4201/v1/',
};
```

6.3.1 Automatisiert hosten

Sofern 6.1.2 installiert wurde, genügt es den folgenden Befehl auszuführen:

```
raspi@raspi: ~/wetterstation$ npm run frontend-prod
```

Es werden folgende Schritte ausgeführt:

- cd frontend
- npm install
- npm run build (siehe 6.3.2)
- npm run host

Anschließend ist das Frontend über die IP-Adresse oder den Namen des Raspberry Pis im selben WLAN-Netzwerk über den Port 8080 erreichbar, z.B. http://192.168.178.190:8080

6.3.2 Angular Build ausführen

Sollte ein andere Http-Server verwendet werden, dann wird der Build-Prozess durch folgenden Befehl angestoßen:

```
raspi@raspi: ~/wetterstation$ cd frontend && npm run build
```

Die Build-Dateien werden anschließend im folgenden Ordner abgelegt:
`~/wetterstation/frontend/dist/wetterstation`