Projektprotokoll – ESP32 Wetterstation

Projektteam: Muhammed Akkus & Mahir Kaplan

Projekt: Entwicklung einer ESP32-basierten Wetterstation mit Webinterface, Sensorik, Status LED und WLAN-Funktionalität

---

Ich habe zusammen mit Mahir Kaplan an der API-Variante der ESP32-Wetterstation gearbeitet. Unsere Aufgabe war es, eine funktionierende Wetterstation zu bauen, die über eine API aktuelle Sensordaten im JSON-Format ausgeben kann. Die Werte sollten Temperatur, Luftfeuchtigkeit und einen weiteren von der Lehrperson gegebenen Sensor enthalten. Außerdem mussten die Daten sinnvoll aufbereitet werden, also mit Mittelwerten und Zeitstempel, und der ESP32 sollte über einen eigenen Access Point konfigurierbar sein. Zusätzlich war eine RGB-LED zu verwenden, um den Status der Wetterstation darzustellen.

Am Anfang haben wir den DHT11-Sensor eingebunden, um Temperatur und Luftfeuchtigkeit zu messen. Um realistische Daten zu bekommen, haben wir den Sensor dreimal hintereinander messen lassen und den Durchschnitt berechnet. Damit keine falschen Werte angezeigt werden, haben wir außerdem Ausreißer ausgeschlossen, z. B. Temperaturen unter -20 °C oder über 60 °C, oder Luftfeuchtigkeit außerhalb von 0 bis 100 %. Der zweite Sensor war bei uns ein Joystick, und wir haben die X-Achse über einen analogen Pin ausgelesen und auf eine Skala von 0 bis 100 umgerechnet.

Für die API haben wir verschiedene Endpunkte erstellt, z. B. /api/data für alle Daten, /api/data/temperature, /api/data/humidity und /api/data/sensor für die Einzelwerte. Alle Daten werden als JSON mit Zeitstempel zurückgegeben. Die Zeit wird über das Internet mit configTime() synchronisiert, und der Zeitstempel wird im ISO-Format angezeigt. Das war wichtig, damit die Daten nicht nur korrekt, sondern auch nachvollziehbar sind.

Ein großes Problem am Anfang war, dass sich der ESP32 nicht richtig mit dem WLAN verbunden hat. Wir haben zuerst autoConnect() vom WiFiManager verwendet, aber der Access Point ist beim ersten Start nicht erschienen. Deshalb haben wir startConfigPortal() verwendet, was viel besser funktioniert hat, weil damit immer ein WLAN mit dem Namen "ESP-Wetterstation" geöffnet wird. Darüber konnten wir den ESP32 direkt ins Heimnetz bringen, ohne manuell etwas im Code ändern zu müssen.

Auch die RGB-LED war ein wichtiger Teil. Wir haben sie so programmiert, dass sie je nach Status verschiedene Farben zeigt: rot, wenn kein WLAN verbunden ist; grün, wenn alles verbunden ist; blau, während gemessen wird; und orange, wenn die Temperatur über 30 Grad steigt. Anfangs hat sich die LED nach dem Ausschalten nicht mehr an die letzte Farbe erinnert. Um das zu lösen, haben wir die zuletzt gesetzte Farbe in drei Variablen gespeichert (R, G, B), und beim Einschalten wurde die gleiche Farbe wieder angezeigt. Außerdem haben wir noch einen Button auf der Startseite eingebaut, mit dem man die RGB-LED per Klick ein- und ausschalten kann. Das hat uns zwar einige Versuche gekostet, weil der Fetch-Befehl in JavaScript am Anfang nicht funktioniert hat, aber wir haben es schließlich gelöst.

Ich habe mich vor allem um den Sensorcode, die Mittelwertberechnung, die LED-Logik und das Webinterface gekümmert, während Mahir die API-Endpunkte strukturiert und getestet hat. Die Zusammenarbeit hat sehr gut funktioniert, weil wir immer gemeinsam getestet und besprochen haben, was noch fehlt oder verbessert werden kann.

Am Ende funktioniert alles so, wie es laut Aufgabenstellung gefordert war. Die Daten können über die API korrekt abgerufen werden, der Status wird über die LED angezeigt, es gibt einen konfigurierbaren Access Point mit WiFiManager und man kann die LED über das Webinterface steuern. Damit haben wir alle Punkte der API-Variante vollständig umgesetzt.

Verfasst von:

Muhammed Akkus

Mahir Kaplan