Projektprotokoll: Wetterstation mit ESP32

Autoren: Marc Reinold & Filip Mladenovic
Datum: 27.05.2025

1. Projektziel

Ziel des Projekts ist der Bau einer intelligenten Wetterstation mit einem ESP32-Mikrocontroller. Die Station misst Temperatur und Luftfeuchtigkeit mit einem DHT11-Sensor und stellt die Daten über ein Webinterface zur Verfügung. Zusätzlich signalisiert eine RGB-LED den Status der Station. Die Messwerte werden mit einem Zeitstempel versehen, der über NTP synchronisiert wird.

2. Verwendete Hardware

Komponente	Beschreibung
ESP32	Mikrocontroller mit WLAN
DHT11	Sensor für Temperatur & Luftfeuchtigkeit
RGB-LED (intern)	Statusanzeige über Farben
HC-SR04 (optional)	Ultraschallsensor für Abstandsmessung

3. Schaltung

- **DHT11: Datenpin an GPIO D2**
- **RGB-LED: Gesteuert über interne GPIOs des ESP32**
- **HC-SR04 (optional): Trigger- & Echo-Pin an frei wählbare GPIOs**

4. Verwendete Bibliotheken

cpp

KopierenBearbeiten

```
#include <WiFi.h>
#include <WiFiManager.h>
#include <ESPAsyncWebServer.h>
#include <DHT.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
```

5. Quellcode und Erklärung

cpp

KopierenBearbeiten

```
// Bibliotheken für Netzwerk, Sensoren und Webserver
#include <WiFi.h>
#include <WiFiManager.h> // einfache WLAN-Verwaltung
#include <ESPAsyncWebServer.h> // Webserver
#include <DHT.h> // Temperatur/Luftfeuchtigkeit
#include <NTPClient.h> // Zeit vom Internet
#include <WiFiUdp.h>

// DHT11-Sensor Setup
#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

// RGB-LED Pins
const int RED_PIN = 15;
const int GREEN_PIN = 4;
```

```
const int BLUE_PIN = 16;

// Webserver läuft auf Port 80
AsyncWebServer server(80);

// NTP-Zeitsetup
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 3600, 60000); // MEZ-Zeitzone (UTC+1)

// Globale Variablen
String temperature = "";
String humidity = "";
String dateTime = "";
bool ledOn = true;

// Hilfsfunktion: RGB-Farbe setzen
void setColor(int red, int green, int blue) {
  analogWrite(RED_PIN, 255 - red);   // invertierte Logik
  analogWrite(GREEN_PIN, 255 - green);
  analogWrite(BLUE_PIN, 255 - blue);
}

// Funktion zum Steuern des Status-LEDs
void updateStatusLed() {
  if (!WiFi.isConnected()) {
    setColor(255, 0, 0); // Rot = kein WLAN
  } else if (temperature.toFloat() > 30) {
    setColor(255, 165, 0); // Orange = zu heiß
```

```
  } else {
    setColor(0, 255, 0); // Grün = alles OK
  }
}

// Sensorwerte lesen
void measure() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();

  if (isnan(temp) || isnan(hum)) {
    Serial.println("Fehler beim Lesen vom DHT11!");
    return;
  }

  temperature = String(temp);
  humidity = String(hum);

  timeClient.update();
  dateTime = timeClient.getFormattedTime();
}

// Setup-Funktion, wird einmalig ausgeführt
void setup() {
  Serial.begin(115200);

  // RGB-LED Pins initialisieren
  ledcSetup(0, 5000, 8); ledcAttachPin(RED_PIN, 0);
```

```
 ledcSetup(1, 5000, 8); ledcAttachPin(GREEN_PIN, 1);
 ledcSetup(2, 5000, 8); ledcAttachPin(BLUE_PIN, 2);

 dht.begin();

 // WLAN konfigurieren
 WiFiManager wm;
 wm.autoConnect("WetterStation");

 timeClient.begin();

 // Webserver-Routen
 server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
  String html = "<html><head><title>Wetterstation</title></head><body>";
  html += "<h1>Wetterdaten</h1>";
  html += "<p>Temperatur: " + temperature + " °C</p>";
  html += "<p>Luftfeuchtigkeit: " + humidity + " %</p>";
  html += "<p>Zeit: " + dateTime + "</p>";
  html += "<form action=\"/toggle\" method=\"POST\">";
  html += "<button type=\"submit\">" + String(ledOn ? "LED ausschalten" : "LED
einschalten") + "</button>";
  html += "</form></body></html>";
  request->send(200, "text/html", html);
 });

 server.on("/toggle", HTTP_POST, [](AsyncWebServerRequest *request){
  ledOn = !ledOn;
  if (ledOn) {
   updateStatusLed();
```

```
  } else {
    setColor(0, 0, 0); // LED aus
  }
  request->redirect("/");
 });

 server.begin();
}

// Endlosschleife
void loop() {
 measure();
 if (ledOn) {
  updateStatusLed();
 }
 delay(5000); // alle 5 Sekunden neue Werte
}
```

6. Webinterface

- **HTML-Seite zeigt Temperatur, Luftfeuchtigkeit und Uhrzeit an.**
- **Button schaltet die RGB-LED ein oder aus.**
- **Keine Neuladung nötig – einfache Steuerung über Web.**

7. Erweiterungsmöglichkeiten

Erweiterung	Status
Farbwahl über Colorpicker	Geplant
Speicherung mit SPIFFS	Geplant
Historie der Messwerte	Optional
Ultraschallsensor integrieren	In Arbeit
Anzeige auf Mobilgeräten	Optimierbar

8. Fazit

Das Projekt kombiniert grundlegende Elemente der IoT-Entwicklung: Sensorik, Netzwerktechnik, Webtechnologien und visuelles Feedback. Die einfache Erweiterbarkeit ermöglicht den Ausbau zur vollständigen Umweltstation.