Wetterstation

Version 1.0

[FIRMENNAME] [Firmenadresse]

Inhaltsverzeichnis

Inhalt

Wetterstation – Projekt, mit ESP32, Datenbank & Display

Fach: Systemtechnik (SYT)

Projektthema: Vernetzte Wetterstation mit ESP32, Sensoren, Display und Webanbindung

Gewählte Varianten: V3: Datenbank + V5: Display

Gruppenmitglieder:

Elias Oncea (Datenbank, LED – Status, Discord – Bot, Verknüpfung)

Enis Shaban (Webserver, Hardware – Bau, Display,)

Datum: 27.05.2025

1) Einleitung:

Selbst gebaute Wetterstationen bieten eine praxisnahe Möglichkeit, Umweltparameter wie Temperatur, Luftfeuchtigkeit oder Lichtintensität in Echtzeit zu erfassen. Der Einsatz solcher Systeme eignet sich besonders im schulischen Umfeld, um grundlegende Kenntnisse in den Bereichen Sensorik, Mikrocontroller-Programmierung und Datenvisualisierung zu vermitteln. In diesem Projekt im Fach Systemtechnik wurde eine eigene Wetterstation mit einem ESP32-C3 Mikrocontroller realisiert. Ziel war es, nicht nur Messdaten zu erfassen und anzuzeigen, sondern auch die gesammelten Daten digital zu verarbeiten, über einen Webserver zugänglich zu machen und stromsparende Konzepte wie den Sleep-Modus zu integrieren. So konnten theoretische Inhalte aus dem Unterricht praktisch angewendet und ein Grundstein für zukünftige IoT-Projekte gelegt werden.

2) Projektbeschreibung:

Das Projekt bestand darin, eine kompakte, internetfähige Wetterstation aufzubauen, die verschiedene Umgebungsdaten messen und ausgeben kann. Der verwendete Mikrocontroller – ein ESP32-C3 – übernimmt die zentrale Steuerung. Über den digitalen Sensor **DHT11** werden Temperatur- und Luftfeuchtigkeitswerte erfasst. Zusätzlich wird ein **Fotowiderstand (B30-Modul)** verwendet, um die Umgebungshelligkeit zu erkennen. Die gemessenen Werte werden auf einem **OLED-Display (SSD1306)** in Echtzeit dargestellt.

Alle erfassten Sensordaten werden zudem in regelmäßigen Intervallen (alle 5–10 Sekunden) in einer Datenbank gespeichert. Die Anbindung an das WLAN-Netz erfolgt über den integrierten WiFi-Chip des ESP32-C3. Mithilfe eines **Webservers**, der direkt auf dem Mikrocontroller läuft, werden die aktuellen Sensorwerte auf einer eigenen Webseite grafisch und textlich dargestellt.

Zur besseren Zustandsanzeige wurde außerdem eine **Status-LED** implementiert, das visuelle Feedback über den Betriebszustand der Wetterstation gibt (z. B. Verbindung erfolgreich, Messung aktiv, Fehlerzustand). Damit das System auch energieeffizient betrieben werden kann, wurde ein **Sleep-Modus** eingebaut, in dem der Mikrocontroller nach einer Datenerfassung für eine bestimmte Zeit in den Energiesparmodus wechselt.

3) Theorie – verwendete Komponenten & Konzepte:

ESP32-C3 Dev Module:
Ein leistungsfähiger Mikrocontroller mit integrierter WiFi-Funktion. Er eignet sich ideal für IoT-Anwendungen und unterstützt moderne Energiesparmodi. Mit seinen GPIOs können verschiedene Sensoren und Module einfach verbunden werden.

DHT11 Sensor:
Ein einfacher digitaler Sensor zur Messung von Temperatur und Luftfeuchtigkeit. Er gibt digitale Werte aus, die direkt vom Mikrocontroller verarbeitet werden können. Er ist kostengünstig, jedoch nicht sehr genau – für schulische Zwecke aber ausreichend.

Fotowiderstand (LDR-Modul B30):
Ein lichtempfindlicher Widerstand, der je nach Lichtintensität seinen Widerstandswert ändert. In der verwendeten Modulversion liefert ein Komparator ein digitales Ausgangssignal (0 oder 1), je nachdem, ob die Lichtintensität über oder unter einem definierten Schwellwert liegt.

OLED-Display (SSD1306, 128x64):
Ein energiesparendes, gut ablesbares Display mit I2C-Schnittstelle. Es eignet sich ideal zur Darstellung von Text und kleinen Grafiken und benötigt nur zwei Datenleitungen (SDA, SCL).

WiFiManager:
Eine Bibliothek zur einfachen Verwaltung von WLAN-Zugängen. Sie erlaubt es, dem ESP32 automatisch bekannte Netzwerke zu verbinden oder über einen Konfigurationsmodus neue Zugangsdaten einzugeben.

Arduino IDE & Bibliotheken:
Die Programmierung des Mikrocontrollers erfolgte über die Arduino-Entwicklungsumgebung. Dabei wurden verschiedene externe Bibliotheken eingebunden, z. B. für den DHT11-Sensor, das OLED-Display und den Webserver.

4) Aufbau & Schaltung:

Alle Komponenten wurden auf einem **Breadboard** aufgebaut und mit dem **ESP32-C3 Dev Module** verbunden. Die Stromversorgung erfolgte über USB oder eine externe 3.3V-Spannungsquelle. Die genaue Pinbelegung war wie folgt:

DHT11 Sensor:

- **VCC** → 3.3V
- **GND** → GND
- **Signal** → GPIO 3

LDR-Modul (B30):

- **VCC** → 3.3V
- **GND** → GND
- **OUT** → GPIO 10

OLED-Display (SSD1306):

- **VCC** → 3.3V
- **GND** → GND
- **SCL** → GPIO 7
- **SDA** → GPIO 6

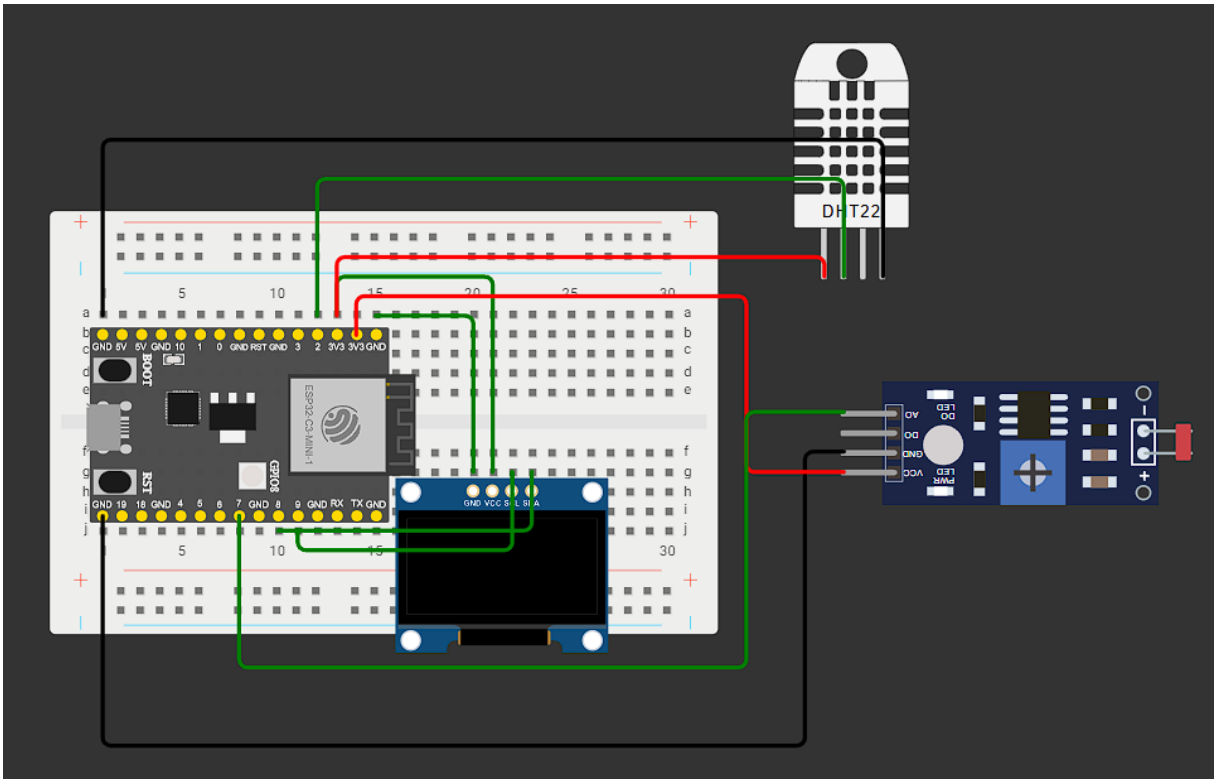
Die genaue Umsetzung mit dem Code folgt unten:

Code:

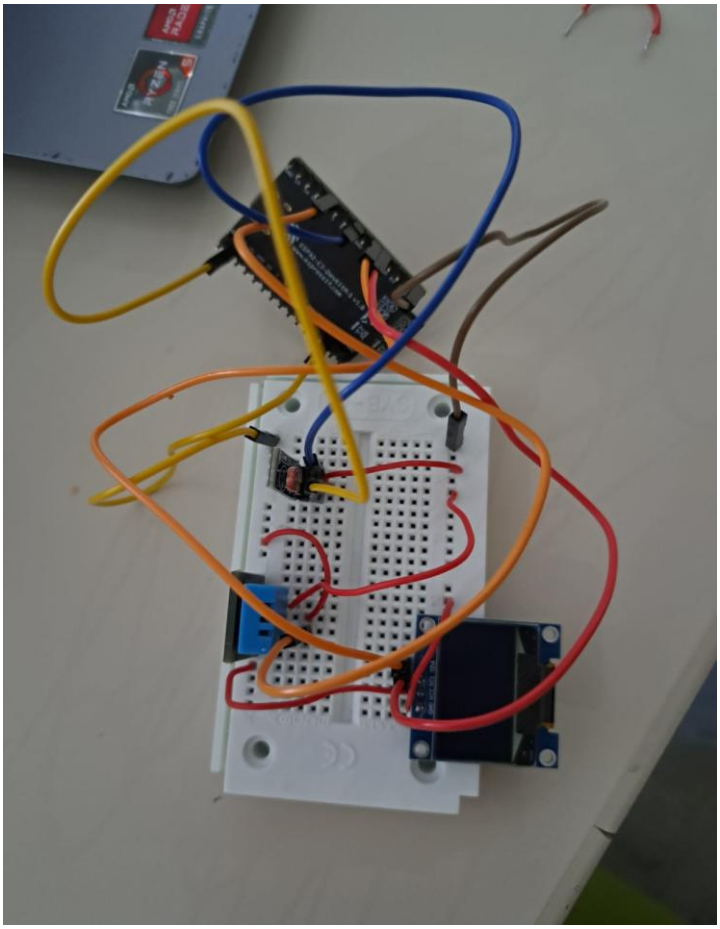
```
#include <DHT.h>                // Bibliothek für DHT-Sensor (Temp./Feuchte)
#include <Adafruit_GFX.h>       // Grafik-Bibliothek für Display-Funktionen
#include <Adafruit_SSD1306.h>   // Bibliothek für OLED-Display (SSD1306)

#define SCREEN_WIDTH 128        // Breite des Displays in Pixel
#define SCREEN_HEIGHT 64        // Höhe des Displays in Pixel
#define OLED_RESET -1           // Reset-Pin (nicht verwendet)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); // Display-Objekt

#define DHTPIN 3                // DHT11 an GPIO 3
#define DHTTYPE DHT11           // DHT-Typ definieren
DHT dht(DHTPIN, DHTTYPE);       // DHT-Objekt initialisieren

#define LDR_PIN 10              // LDR an GPIO 10

void setup() {
 Serial.begin(115200);          // Serielle Schnittstelle starten
 dht.begin();                   // DHT-Sensor starten
 pinMode(LDR_PIN, INPUT);       // LDR-Pin als Eingang

 if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {  // Display starten (Adresse 0x3C)
  Serial.println(F("Display konnte nicht initialisiert werden"));
  for (;;);                     // Endlosschleife bei Fehler
 }
 display.clearDisplay();        // Display löschen
 display.setTextSize(1);        // Textgröße einstellen
 display.setTextColor(SSD1306_WHITE); // Textfarbe: Weiß
}

void loop() {
```

```
  float temp = dht.readTemperature(); // Temperatur vom DHT11 lesen
  float hum = dht.readHumidity();    // Luftfeuchte vom DHT11 lesen
  int licht = digitalRead(LDR_PIN);  // LDR-Wert lesen (hell/dunkel)

  display.clearDisplay();          // Display löschen
  display.setCursor(0, 0);         // Cursor oben links setzen
  display.print("Temp: ");
  display.print(temp);             // Temperatur anzeigen
  display.println(" C");
  display.print("Luftfeuchte: ");
  display.print(hum);              // Luftfeuchte anzeigen
  display.println(" %");
  display.print("Licht: ");
  display.println(licht == 1 ? "Hell" : "Dunkel"); // Lichtstatus anzeigen
  display.display();               // Display aktualisieren

  delay(2000);                     // 2 Sekunden warten
} (Code individuell beliebig anpassbar)
```

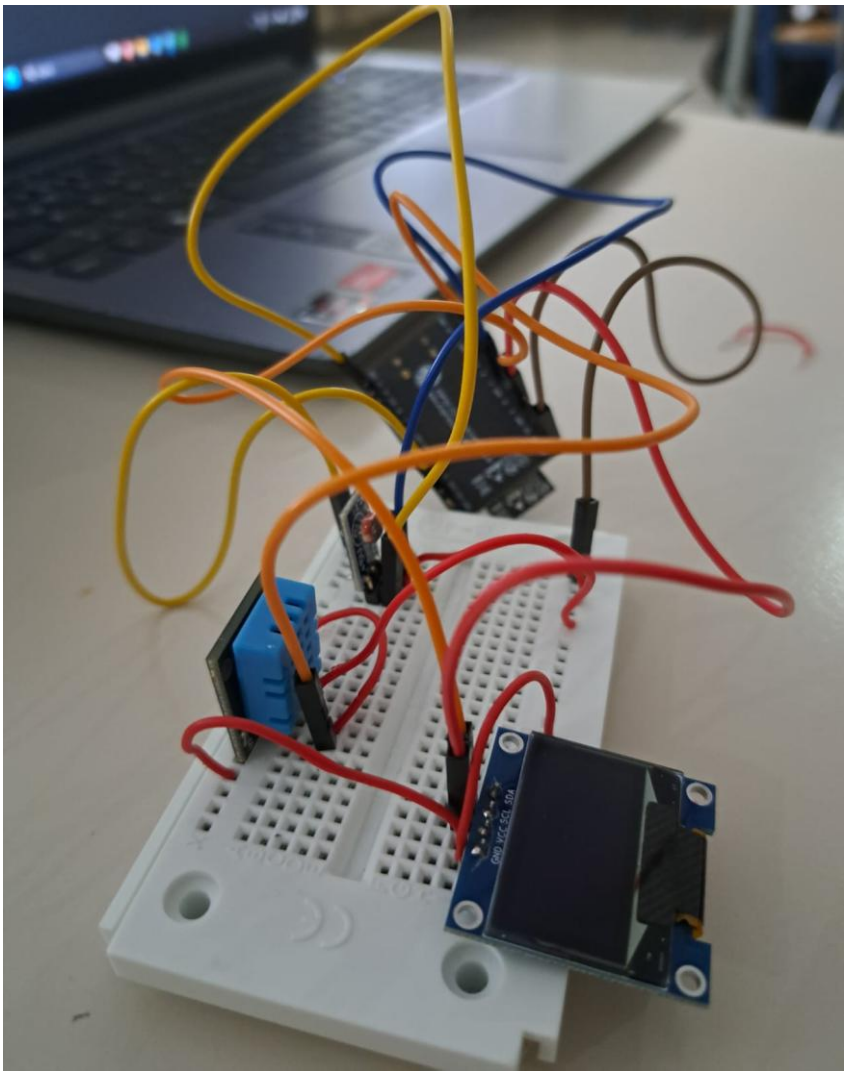
Bilder & Schaltungen:



Physische Schaltung:



Zweite Perspektive:



1.ESP32 C3: Hier läuft die Software, hier wird alles gesteuert. Er ist per USB am Computer angeschlossen und mit dem Breadboard verbunden.

2. DHT22 (Temperatur- und Luftfeuchtigkeitssensor)

- VCC → 3.3 V am ESP32
- GND → GND
- DATA → Pin IO3 (ganz oben rechts am ESP32)

Der Sensor misst regelmäßig Temperatur und Luftfeuchtigkeit und sendet die Daten per Funk zum ESP32.

3. Lichtmodul (Fotowiderstand, LDR B30)

- VCC → 3.3 V
- GND → GND
- DO (digital out) → Pin IO10

Das Modul erkennt, ob Sonne scheint oder es dunkel ist. Es liefert entweder 0 oder 1, je nachdem ob es heller oder dunkler als ein eingestellter Schwellenwert ist (einstellbar mit dem blauen Poti am Modul).

4. OLED-Display (SSD1306, 128x64 Pixel, I2C)

- VCC → 3.3 V
- GND → GND
- SCL → Pin IO7
- SDA → Pin IO6

Das Display zeigt die gemessenen Werte der Sensoren an. An den ESP32 angeschlossen ist es über das I2C-Protokoll sehr schnell und benötigt nur zwei Datenleitungen.

Genauer Betrieb

- Der ESP32 startet und verbindet sich mit Strom.
- Er liest die Daten vom DHT22 und vom Lichtmodul aus.
- Die Werte werden direkt auf dem OLED-Display sichtbar.
- Optional (wenn programmiert): Die Daten werden zusätzlich gespeichert oder per WLAN verschickt.
- Nach einer gewissen Zeit (z. B. 10 Sekunden) wechselt der ESP32 in den Schlaf-(Sleep-)Modus und wacht dann für die nächste Messung wieder auf.

Technische Dokumentation: ESP32-C3 Wetterstation

1. Einleitung

Im Fach Systemtechnik haben wir eine Wetterstation entwickelt, die mit Hilfe eines Mikrocontrollers Umweltinformationen sammelt, verarbeitet und lokal speichert sowie online bereitstellt. Wir hatten uns zum Ziel gesetzt, praxisnah die Sensorik, die Programmierung des Mikrocontrollers, die Darstellung von Daten und energieeffizientes Design kennenzulernen.

2. Projektidee

Das System basiert auf dem Mikrocontroller ESP32-C3, der eine WLAN-Funktion bereits integriert hat. Wir erfassen:

- Temperatur & Luftfeuchtigkeit: DHT11-Sensor
- Lichtintensität: Fotowiderstand (LDR), der je nach Helligkeit seinen Widerstand ändert

Dargestellt werden die Messergebnisse auf einem OLED-Display (SSD1306) und die Ansteuerung erfolgt per I2C-Bus. Unsere Station speichert die Daten zudem regelmäßig in einer Datenbank und stellt sie per integriertem Webserver bereit. Eine Status-LED gibt zudem Auskunft über den aktuellen Betriebszustand der Station. Um möglichst energieeffizient zu arbeiten, wird der ESP32-C3 zwischen den Messungen in den Sleep-Modus versetzt.

3. Grundlagen

- ESP32-C3: preiswerter, stromsparender Mikrocontroller mit WLAN und Bluetooth, ideal für Anwendungen aus dem Internet of Things (IoT).
- DHT11-Sensor: misst Temperatur und Luftfeuchtigkeit, die Datenübertragung erfolgt seriell und einfach.
- LDR: Wird der Widerstand, je nachdem wie hell, ändert eignet sich so ein LDR gut für die Helligkeitsmessung.
- OLED-Display (SSD1306): einfarbig, 128x64 pixelig, stromsparend, gut ablesbar, und per I2C anzubinden.
- WiFiManager: Damit ist die Konfiguration und erste Kozeptur wiedermal einfacher geworden.

4. Aufbau & Schaltung

Als erstes werden hier alle Bauteile miteinander auf einem Breadboard verkabelt. Versorgt wird das ganze über USB. Mittels WLAN wurden die Werte des Sensors erfasst, in unserem Fall fand die Auswertung statt und wurde auf einem angeschlossenen Display angezeigt.

5.Fazit

Dieser kleine Artikel gibt einen Überblick über verschiedene Sensoren und deren Ansteuerung und Auswerteung in Kombination mit einem Display. Bis hin zu den ganzen Funktionen.

6. Quellenverzeichnis

Wikipedia: Technische Dokumentation. Abgerufen am 28. Mai 2025, von https://de.wikipedia.org/wiki/Technische_Dokumentation

Components101: DHT11–Temperature and Humidity Sensor. Abgerufen am 28. Mai 2025, von https://components101.com/sensors/dht11-temperature-sensorComponents101

SunFounder: Fotowiderstands-Modul. Abgerufen am 28. Mai 2025, von https://docs.sunfounder.com/projects/ultimate-sensor-kit/de/latest/components_basic/07-component_photoresistor.htmldocs.sunfounder.com+2docs.sunfounder.com+2docs.sunfounder.com+2

Adafruit: SSD1306 Datasheet. Abgerufen am 28. Mai 2025, von https://cdn-shop.adafruit.com/datasheets/SSD1306.pdfcdn-shop.adafruit.com

Draeger IT Blog: ESP32-C3: Der kleine Alleskönner für deine IoT-Projekte. Abgerufen am 28. Mai 2025, von https://draeger-it.blog/esp32-c3-der-kleine-alleskoenner-fuer-deine-iot-projekte/Technik Blog+1Technik Blog+1