Erinnerung an deinen Regenschirm mit dem ESP8266

Hast du schon einmal deine Wohnung verlassen und bist dann auf der Straße im Regen gestanden? Dafür, dass das nicht noch einmal passiert, sorgst du mit diesem Projekt.

Mit einem ESP8266 rufst du über eine API die aktuelle Wetterlage ab und zeigst auf einem OLED-Display an, ob es regnet, schneit oder die Sonne scheint. **Falls du draußen einen Regenschirm benötigst, erinnert dich das Display und eine LED daran, deinen Regenschirm mitzunehmen.**

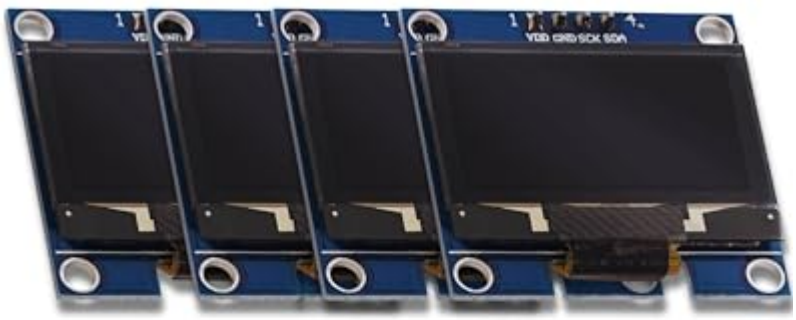
Ach ja: Damit dein ESP8266 nicht den ganzen Tag im Internet verbringt und ständig die Wetterdaten abruft, integrierst du noch eine Radar-Modul. So kannst du dein Projekt zum Beispiel an der Wohnungstür platzieren und es nur dann anspringen lassen, wenn du dich ihm näherst. So wirst du genau zur richtigen Zeit erinnert.

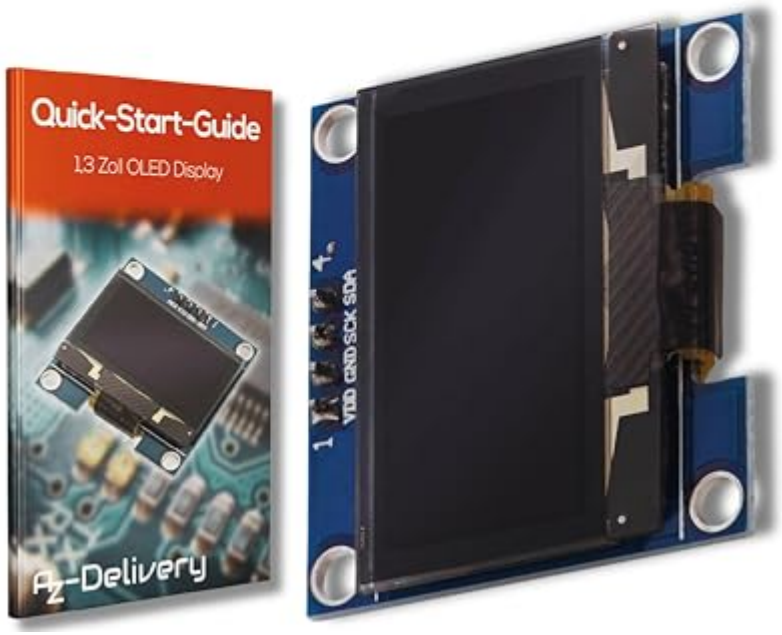
Für dieses Projekt benötigst du (Mengen s. Beschreibung):

[AZDelivery NodeMCU Amica Modul V2 ESP8266 ESP-12F WiFi - Node MCU ESP 8266 WiFi Development Board mit CP2102 kompatibel mit Arduino - inklusive Installationsanleitung als E-Book]

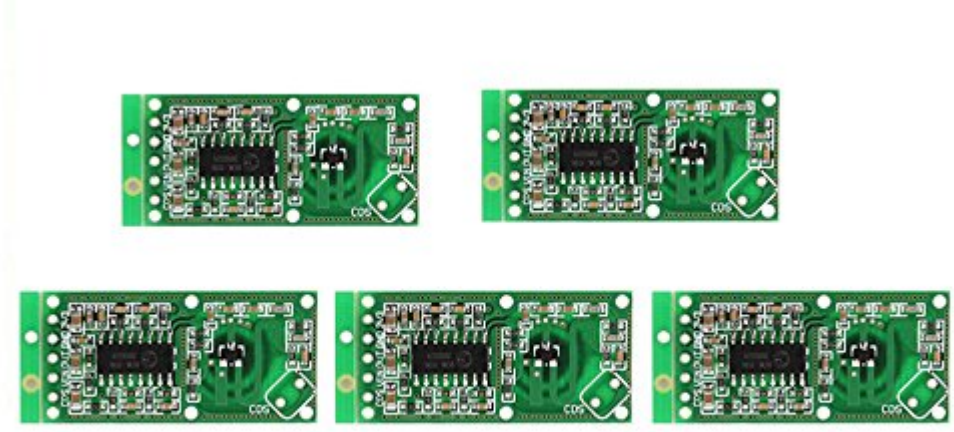
 Maße (LxBxH): 48 x 26 x 13 mm

8,49 €

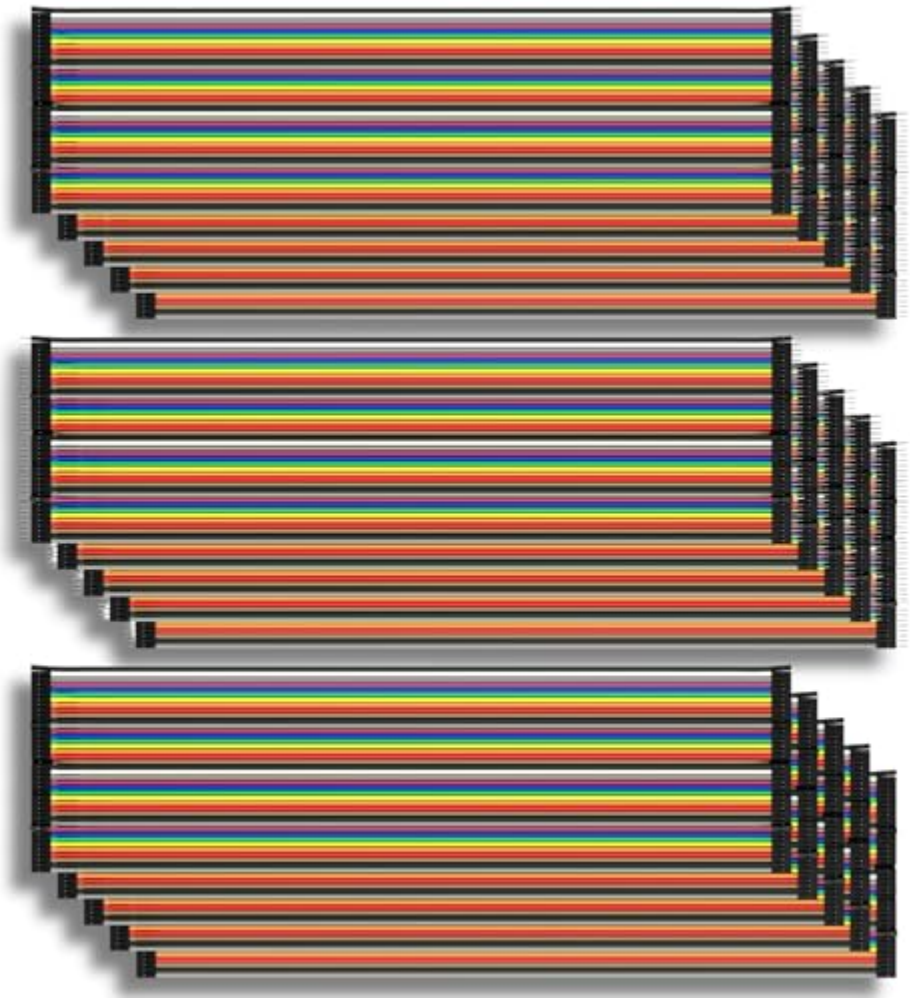




[AZDelivery 1 x 1,3 Zoll OLED Display I2C SSH1106 Chip 128 x 64 Pixel I2C Bildschirm Anzeigemodul mit weißen Zeichen | I2C Display ssd1306 | kompatibel mit Arduino und Raspberry Pi inklusive E-Book!]
8,49 €



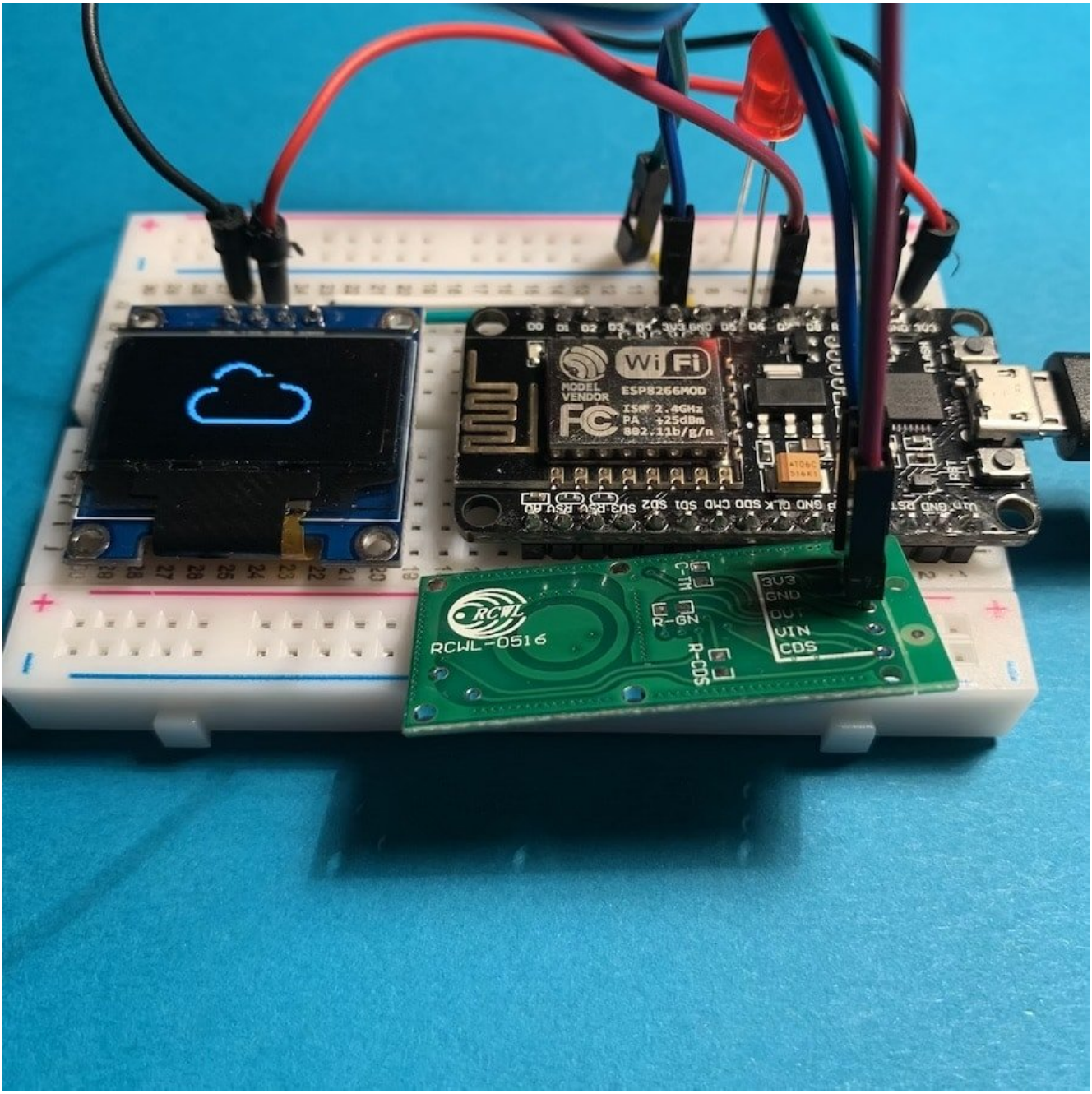
[WINGONEER 5PCS Mikrowellen-Radarsensor RCWL-0516 Schalter-Modul menschlicher Induktions-Brett-Detektor]
4,99 €

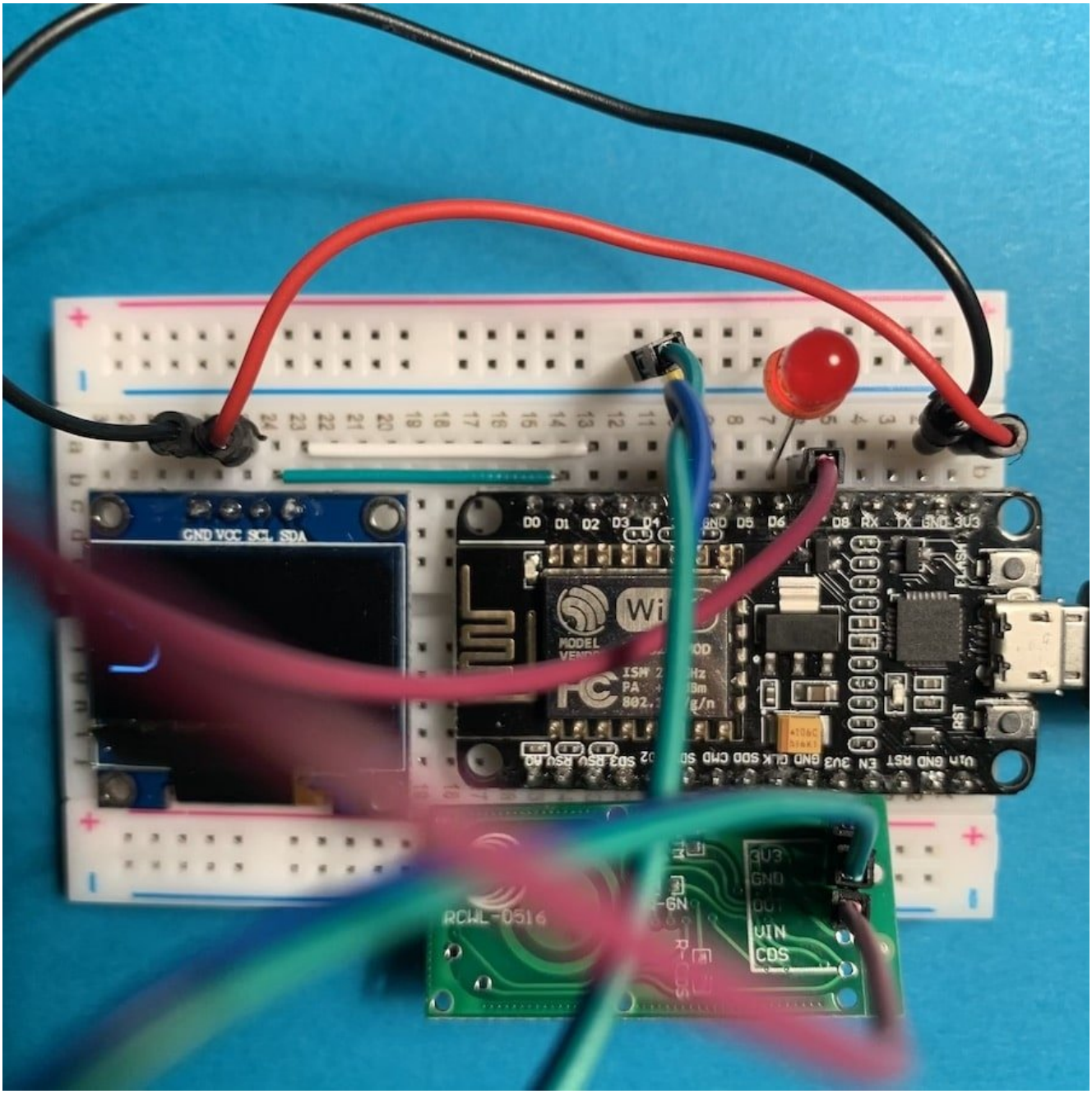


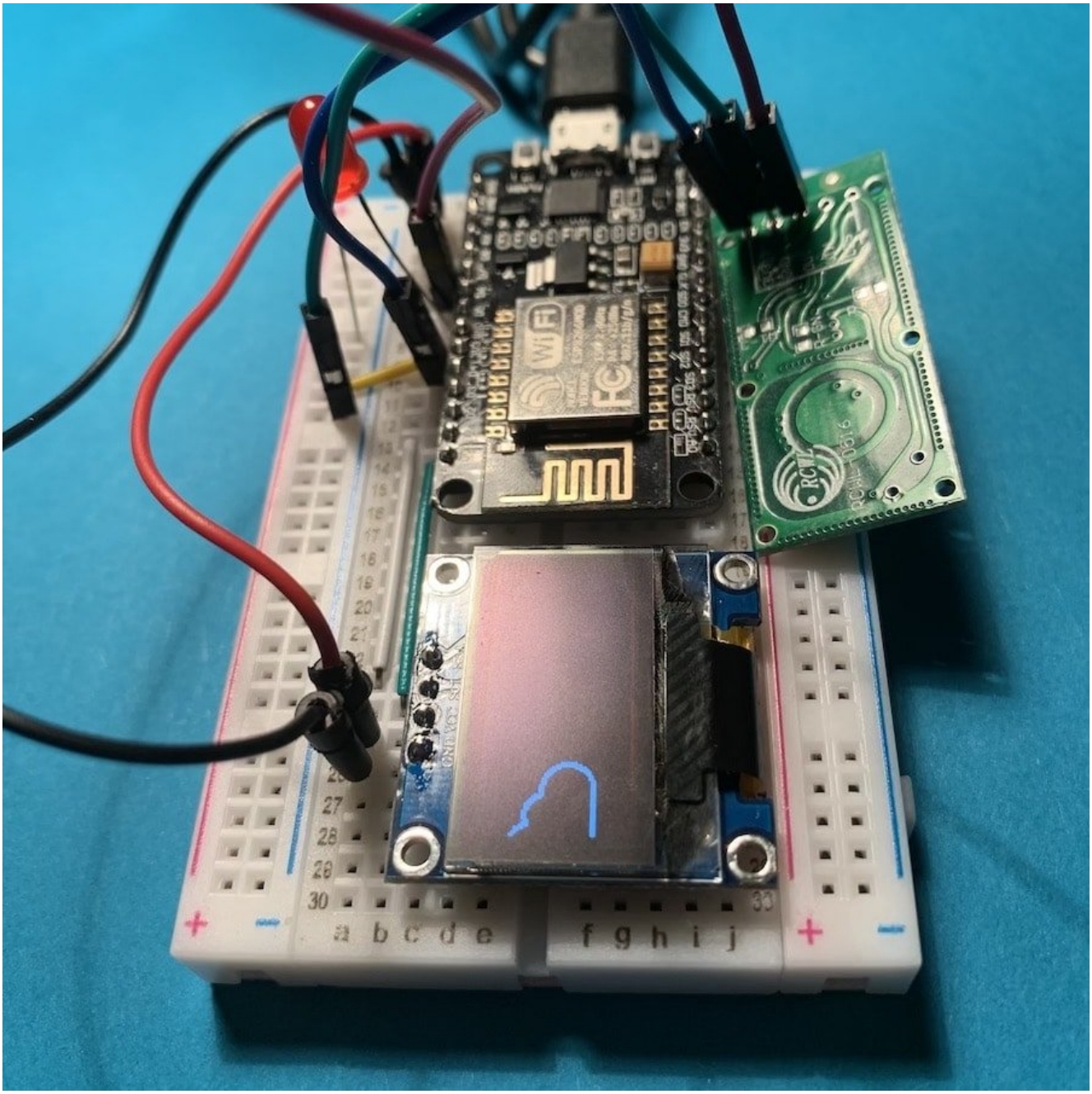
[AZDelivery Jumper Wire Kabel 3 x 40 STK. je 20 cm M2M/ F2M / F2F kompatibel mit Arduino und Raspberry Pi Breadboard inklusive E-Book!]
6,99 €

Der Aufbau

So ungefähr sollte das fertig aufgebaute Projekt aussehen:


Wi Fi
MODEL
VENDOR
ESP8266MOD
ISM 2.4GHz
PA +25dBm
802.11b/g/n
RCWL-0516
3V3
GND
OUT
VIN
CDS


GND VCC SCL SDA
D0 D1 D2 D3 D4
D8 RX TX GND 3V3
RCWL-0516
3V3
GND
VIN
CDS



Setze zuerst deinen ESP8266 auf dein Breadboard. Ich empfehle hierfür ein **NodeMCU Lua Amica**, denn das ist schmal genug, dass du an den Pinouts noch Kabel einstecken kannst. Die neuere Version namens **Lolin** ist da im Nachteil und deshalb für Projekte auf Breadboards eher ungeeignet. Anschließend verbindest du deinen Microcontroller mit diesem 3 Bauteilen:

OLED-Display

In diesem Projekt verwendest du ein kleines OLED-Display mit

I²C-Anschluss. Setze dein Display auf das Breadbaord und verbinde es zunächst mit einem der GND-Pins und einem 3,3V-Ausgang am Microcontroller. Anschließend verbindest du den Display-Pin **SCL mit D1** und den Pin **SDA mit D2**. Mit den beiden Pins D1 und D2 kannst du per I²C mit deinem Display kommunizieren.

Radar-Modul RCWL-0516

Dieser Sensor ist ein einwandfreier Bewegungssensor mit einer Reichweite von 5 bis 7 Metern. Das Tolle an diesem Modul ist, dass es sogar Bewegungen durch ein Hindernis (zum Beispiel eine Box) erkennen kann.

Der Anschluss ist kinderleicht: Neben dem üblichen Anschluss von GND und 3,3V verbindest du den **Pin OUT mit dem Pin D7** an deinem ESP8266.

Die LED

Nun zur allerleichtesten Übung, der LED. Verbinde die Anode (langes Bein) mit dem Pin D6 und die Kathode (kurzes Bein) mit Erde. Vergiss nicht, einen Widerstand zwischenzuschalten!

Der Code für dein Projekt

Etwas komplexer wird es jetzt, denn zunächst musst du dich für eine **-kostenlose-** API registrieren, von der du die aktuelle Wetterlage an deinem Wohnort (oder an jedem anderen Ort der Erde) abrufen kannst.

Registriere dich zunächst auf https://openweathermap.org. Für einen Account benötigst du nur einen Usernamen, eine E-Mail-Adresse und ein Passwort. In deinem persönlichen Bereich hast du unter dem Menüpunkt **API keys** nun die Möglichkeit einen solchen API key zu erstellen. Diesen benötigst du später in deinem Arduino Sketch für die API-Abfrage der Wetterdaten. Wenn du das gemacht hast, kann es mit dem Code weitergehen.

Die benötigten Bibliotheken

Für dieses Projekt benötigst du 7 Bibliotheken. 3 hiervon musst du evtl. erst noch installieren, um sie nutzen zu können: **Adafruit GFX, Adafruit 1306 und ArduinoJSON**. Falls du sie bisher in deiner Arduino IDE noch nicht installiert hast, hole das am besten gleich nach. Klicke hierfür auf den Menüpunkt **Werkzeuge** und anschließend auf **Bibliotheken verwalten**. Anschließend öffnet sich der sogenannte **Bibliotheksverwalter**, in dem du nach den jeweiligen Bibliotheken suchen kannst. Installiere jeweils die neueste Version.

Wenn du damit fertig bist, bindest du sie noch vor der Funktion **void setup()** am Anfang deines Sketchs ein:

```
#include <Wire.h> //I2C-Verbindung
#include <Adafruit_GFX.h> //OLED-Display
#include <Adafruit_SSD1306.h> //OLED-Display

#include <ESP8266WiFi.h> //WiFI
#include <ArduinoJson.h> //JSON
#include <ESP8266HTTPClient.h> //API-Abfrage
#include <WiFiClient.h> //API-Abfrage
```

Zugangsdaten zum Internet und OLED-Display

Damit sich dein ESP8266 mit dem Internet verbinden kann, musst du deine Zugangsdaten hinterlegen. Gleich dahinter folgt etwas Code für die Kommunikation mit deinem Display:

```
// WIFI-Zugangsdaten
const char* ssid = "Dein Netzwerkname";
const char* password =  "Dein Passwort";


//OLED-Display
#define SCREEN_WIDTH 128 // Breite des Displays in Pixeln
#define SCREEN_HEIGHT 64 // Höhe des Displays in Pixeln
```

```
//Angaben zum OLED-Display und der Kommunikation per I2C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
-1);
```

Icons für alle (wichtigen) Wetterlagen

Auf deinem Display sollen später Icons erscheinen, die das aktuelle Wetter symbolisieren: Klarer Himmel, Wolken, Schnee, Nebel – und ein Regenschirm für Regen und Gewitter. Die Deklarationen dieser Icons im Code sind sehr lang. Deshalb findest du sie nicht an dieser Stelle, sondern im gesamten Arduino-Sketch, den du unten findest.

Die Setup-Funktion

In der Funktion **void setup()** definierst du die beiden Pins für die LED und den Radar-Sensor, bereitest du die Verbindung zum Seriellen Monitor vor, gibst die Adresse deines Displays an und legst die Schriftgröße und -farbe fest.

```
void setup() {

  pinMode (12, OUTPUT); //LED Pin (am ESP8266 D6)
  pinMode (13, INPUT); //Radar Pin (am ESP8266 D7)

  Serial.begin(115200); //Verbindung zum Seriellen Monitor

   if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { //
Display-Addresse: 0x3C
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }

  display.setTextSize(1); //Schriftgröße
  display.setTextColor(SSD1306_WHITE); //Farbe
  display.clearDisplay();
}
```

Der Loop

In der Funktion **void loop()** arbeitet dein Radar-Modul. Alle 100 Millisekunden prüft es, ob sich etwas in der Nähe bewegt. Ist das der Fall, verbindet sich dein ESP8266 mit dem Internet und startet die Abfrage der Wetterdaten von Openweather. Solange das nicht der Fall ist, läuft der Loop immer weiter und das OLED-Display bleibt dunkel.

```
void loop() {
  delay(100);
  if (digitalRead(13) == HIGH) { //Bewegung erkannt?
    Serial.println("Movement detected.");
    if (WiFi.status() != WL_CONNECTED) { //Nicht verbunden -> Verbindung herstellen
      WiFi.begin(ssid, password);

      while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        display.clearDisplay();
        display.setCursor(0, 10);
            display.println("Connecting to WiFi..."); //Verbindungsstatus anzeigen
        display.display();
      }

      display.clearDisplay();
      display.setCursor(0, 10);
             display.println("Connected to WiFi!"); //Verbindungsstatus anzeigen
      display.display();
      delay(1000);
    }

    apiCall(); //Funktion apiCall() aufrufen
  } else { //Keine Bewegung erkannt?
    if (WiFi.status() == WL_CONNECTED) { //dann die bestehende Verbindung beenden
      WiFi.disconnect();
      Serial.println("Disconnected from WiFi");
    }
```

```
    digitalWrite(12, LOW);
  }
}
```

Die Funktion apiCall()

Diese Funktion ist das Herzstück deines Codes! Hier startest du die Abfrage bei der API (ein sogenannter API Call), verarbeitest die in deinem ESP8266 ankommenden Daten und zeigst abhängig vom Wetter ein entsprechendes Icon auf deinem Display an. Zusätzlich gibst du noch die aktuelle Temperatur aus und lässt bei Regen die LED leuchten.

Dieser Code ist nicht ganz einfach. Deshalb schauen wir ihn uns Schritt für Schritt an:

Als allererstes definierst du natürlich die Funktion selbst und schreibst eine If-Abfrage, innerhalb der sich der gesamte folgende Code abspielt. Mit dieser Abfrage stellst du sicher, dass dieser nur ausgeführt wird, wenn die Verbindung zum Internet steht. Denn ohne Internet keine Wetterdaten.

```
void apiCall() {

  if ((WiFi.status() == WL_CONNECTED)) { //Netzwerkstatus checken
```

Wetterdaten abrufen und verarbeiten

Nun kommt die Abfrage bei Openweather. Hierfür verwendest du eine URL, in der die Art der Abfrage (aktuelles Wetter) und der gewünschte Ort definiert sind. Am Ende der URL steht dein API key, den du zu Beginn erstellt hast. Nähere Informationen erhältst du in der Dokumentation von Openweather.

Eine mögliche URL für die Abfrage des Berliner Wetters könnte so aussehen:

```
http://api.openweathermap.org/data/2.5/weather?q=Berlin,de&units=metric&appid=DEIN API KEY"
```

Trage hinter **q=** die Stadt ein, deren Wetter du haben möchtest. Alternativ kannst du auch die Koordinaten deines Standorts verwenden. Um diese herauszufinden, eignet sich die Webseite geoplaner.de – dort kannst du einen beliebigen Ort eingeben und dessen Koordinaten herausfinden (im gelben Kasten). Für Berlin würde die URL dann wie folgt lauten:

```
http://api.openweathermap.org/data/2.5/weather?lat=52.51&lon=13.39&units=metric&appid=DEIN API KEY"
```

Jetzt folgt ein Check, ob die Abfrage funktioniert hat. In diesem Fall erhältst du vom Server die Antwort **200**. Also schreibst du eine If-Abfrage, die allen folgenden Code nur ausführt, wenn die Server-Antwort positiv ausgefallen ist und du die gewünschten Daten erhalten hast.

Danach speicherst du diese Daten in der Variable **payload** und verarbeitest sie mit Hilfe der Bibliothek ArduinoJson. Die Erklärung, wie das genau funktioniert, würde den Rahmen hier sprengen. Nähere Erläuterungen findest du im Projekt Ein Newsticker per API Call & JSON und auf der Webseite des Entwicklers dieser Bibliothek.

```
    if (httpCode == HTTP_CODE_OK) { //Ist die Antwort des Servers 200?

       String payload = http.getString(); //Daten in eine Variable speichern
      DynamicJsonDocument doc(1024);

        DeserializationError error = deserializeJson(doc, payload); //JSON parsen

        if (error) { //Fehlermeldung bei fehlerhafter Verarbeitung
        Serial.print(F("deserializeJson() failed: "));
        Serial.println(error.c_str());
```

```
        return;
      }
```

In der Variable **payload** befinden sich nun die Wetterdaten im JSON-Format. Wie diese aussehen, kannst du sehen, wenn du die URL zur Abfrage aus deinem Sketch in deinem Browser aufrufst. Hier ein Beispiel für Berlin am Abend des 13. Januar 2020:


```
{"coord":{"lon":13.41,"lat":52.52},"weather":[{"id":802,"main":"Clouds","description":"scattered clouds","icon":"03n"}],"base":"stations","main":{"temp":3.2,"feels_like":-1.49,"temp_min":1.11,"temp_max":5,"pressure":1017,"humidity":86},"visibility":10000,"wind":{"speed":4.1,"deg":180},"clouds":{"all":40},"dt":1578947701,"sys":{"type":1,"id":1275,"country":"DE","sunrise":1578899517,"sunset":1578928632},"timezone":3600,"id":2950159,"name":"Berlin","cod":200}
```


Etwas schwer zu lesen, oder? □ Für deinen ESP8266 ist das jedoch kein Problem. Die Bibliothek ArduinoJson stellt dir hierfür alles zur Verfügung. Kurz gesagt, kannst du mit einer paar Zeilen Code genau die Daten herausfischen, die du brauchst. Über allem steht ein **For-Loop**, der dafür sorgt, das die kommenden Anzeigen und Animationen drei Mal angezeigt werden, bis das Display wieder erlischt:

```
      //Wetterlage und Temperatur ausgeben
        for (int j = 0; j < 3; j++) { // 3x Animation durchlaufen lassen bis zur nächsten Messung
        //Wetterdaten auslesen anhand von ID
        JsonObject weather_0 = doc["weather"][0];
        int weather_0_id = weather_0["id"]; // Wetter-ID
        Serial.println(weather_0_id);
```

Aus den Daten hast du jetzt die ID des aktuellen Wetters herausgelesen. Openweather vergibt nämlich jeder Wetterlage eine eigene dreistellige ID. So hat leichter Regen zum Beispiel die ID 500. Herabrieselnde Vulkanasche hingegen die

ID 762! Alle IDs findest du hier.

Wetterdaten auf dem Display zeigen

Jetzt folgen eine Reihe von If-Abfragen, die das zum Wetter passende Icon auf deinem Display zeigen und animieren. Hier exemplarisch der ID-Bereich „Wolken“ – das sind alle IDs zwischen 801 und 804. Nähere Erläuterungen findest du in den Kommentaren im Code. Interessant ist der For-Loop: Hier verschiebst du das Icon nach links, indem du es 128 Mal (die Breite des Icons) immer einen Pixel weiter links anzeigst. Das erzeugt den Eindruck einer Animation.

```
        if (weather_0_id > 800 && weather_0_id < 900) {
          //Cloud anzeigen und wegschieben
          display.clearDisplay(); //Display löschen
           display.drawBitmap(0, 0, clouds, 128, 64, WHITE);
//Icon zeichnen...
          display.display(); //...und anzeigen
           digitalWrite(12, LOW); //LED ausschalten, da kein
Regen
          delay(2000);
           for (int i = 0; i > -128; i--) { //Das Icon aus dem
Display "schieben"
             display.clearDisplay();
             display.drawBitmap(i, 0, clouds, 128, 64, WHITE);
             display.display();
             delay(1);
          }
```

Es folgen weitere Abfragen per **else if{}** für die anderen Wetterlagen, die du unten im gesamten Sketch findest.

Jetzt fehlt noch die Anzeige der Temperatur. Auch diese liest du aus den JSON-Daten aus und bringst sie animiert auf dein Display:

```
//Temperatur anzeigen und animieren
        JsonObject main = doc["main"];
        int main_temp = (int)main["temp"]; // Daten in ein INT
```

```
umwandeln und in Variable speichern
         String temp = String(main_temp) + " C"; //C für
Celsius anhängen
        display.setTextSize(3); //Schriftgröße erhöhen
        display.setCursor(25, 20);
        display.println(temp);
        display.display();
        delay(2000);
        for (int i = 25; i > -144; i--) { //Animation
          display.clearDisplay();
          display.setCursor(i, 20);
          display.println(temp);
          display.display();
          delay(1);
        }
```

Zuletzt setzt du die Schriftgröße wieder auf 1 und schließt alle offenen Klammern. Das allerletzte **else** bezieht sich auf die Abfrage, ob die Antwort des Servers positiv (also 200) war und gibt eine Fehlermeldung zurück, falls das nicht so war.

Hier nun der vollständige Sketch. Wenn du ihn kopierst und verwendest, achte unbedingt darauf, deine **eigenen WLAN-Daten und deinen API key von Openweather in die markierten Stellen einzutragen**.

Sketch auf Github ansehen

Wie geht es weiter?

Du hast jetzt ein Gerät gebaut, das dich an deinen Regenschirm erinnert. Integriere es in eine Box und versorge den ESP8266 zum Beispiel mit einer Powerbank über USB mit Strom. Diese Box kannst du in der Nähe deiner Wohnungstür platzieren, damit

dich dein Projekt genau im richtigen Moment vor dem Regen draußen warnt.

Du denkst größer und an deinen eigenen Wetter-Server? Dann lerne hier, wie du deinen ESP8266 in einen Web Server verwandelst. Oder wie wäre es mit einer ESP8266 Wetterstation, die Daten auf einem Raspberry Pi sammelt und visualisiert?

Beachte jedoch: Zu 100% kannst du dich leider nicht auf die Wetterdaten aus dem Internet verlassen. Manchmal ist vielleicht ein prüfender Blick aus dem Fenster doch sicherer.