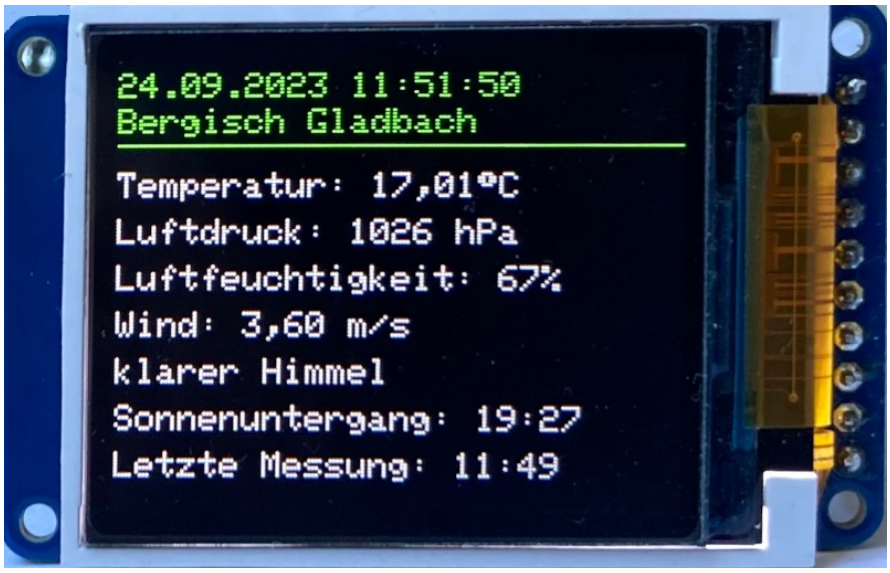
Inhaltsverzeichnis

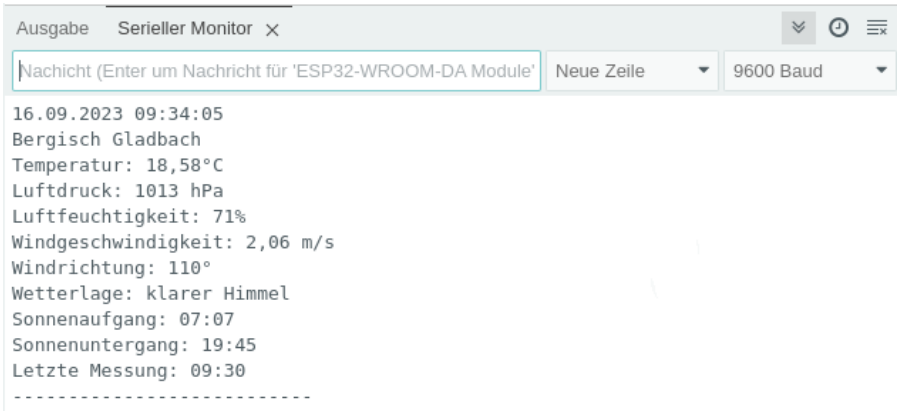
Mit der API (Application Programming Interface = Programmierschnittstelle) von Openweathermap.org und den damit erhobenen Daten sollen die Wetterdaten auf einem TFT-Display und in etwas ausführlicherer Form im Seriellen Monitor angezeigt werden.

Darstellung der Daten

TFT-Display



Serieller Monitor



Vorbereitung

Zunächst benötigst du einen API-Schlüssel von Openweathermap.org:

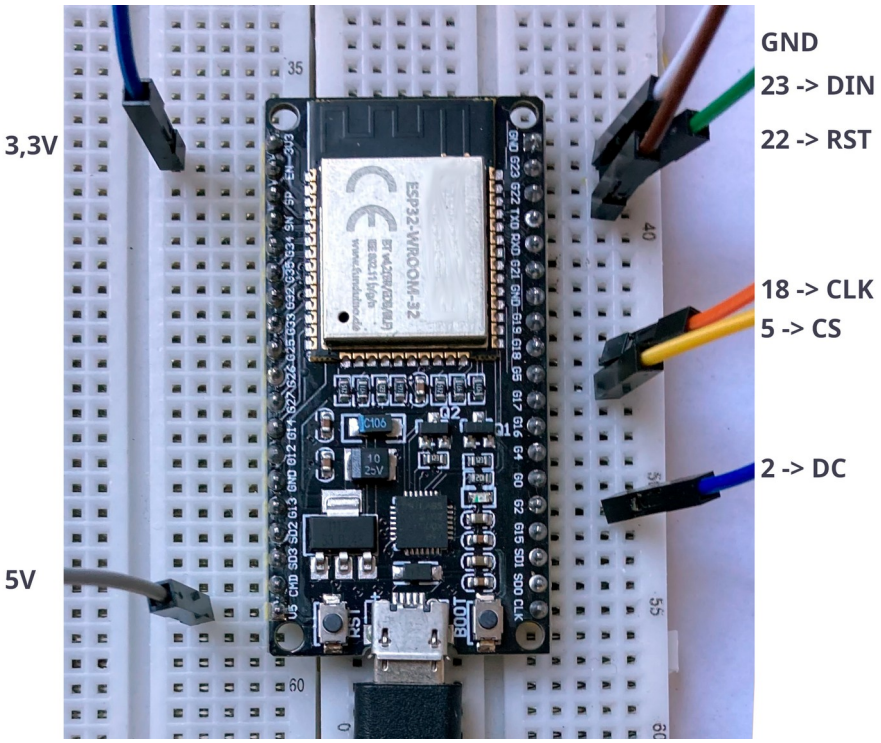
https://home.openweathermap.org/users/sign_up

In der Bestätigungsmail findest du deinen API-Key. Der freie Zugang erlaubt 60 Zugriffe in der Minute.
Quelle: https://openweathermap.org/price#weather

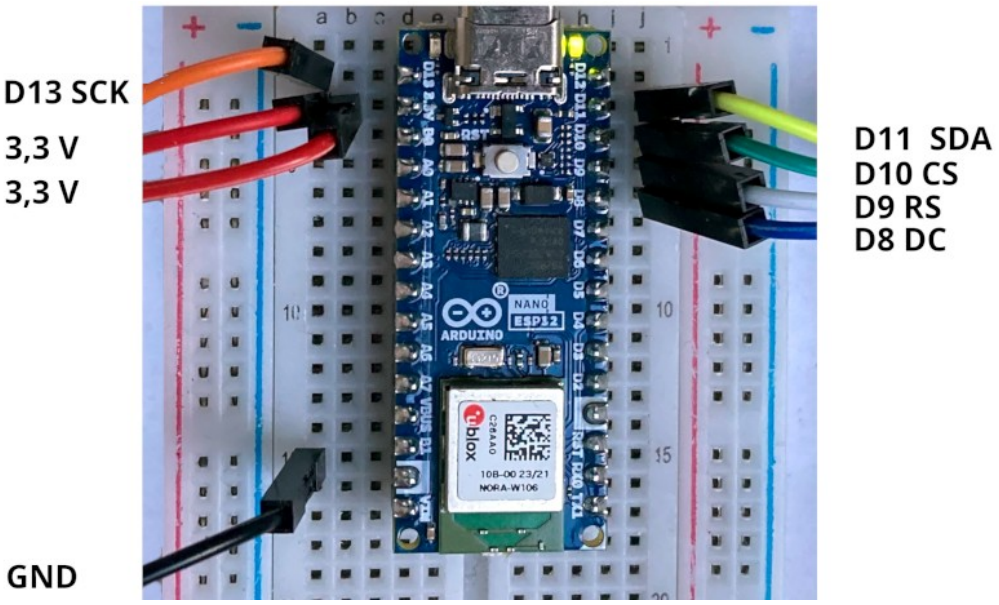
Benötigte Bauteile

- ESP32-Wroom oder Arduino Nano ESP32
- TFT
- Leitungsdrähte

Pinbelegung ESP32-Wroom



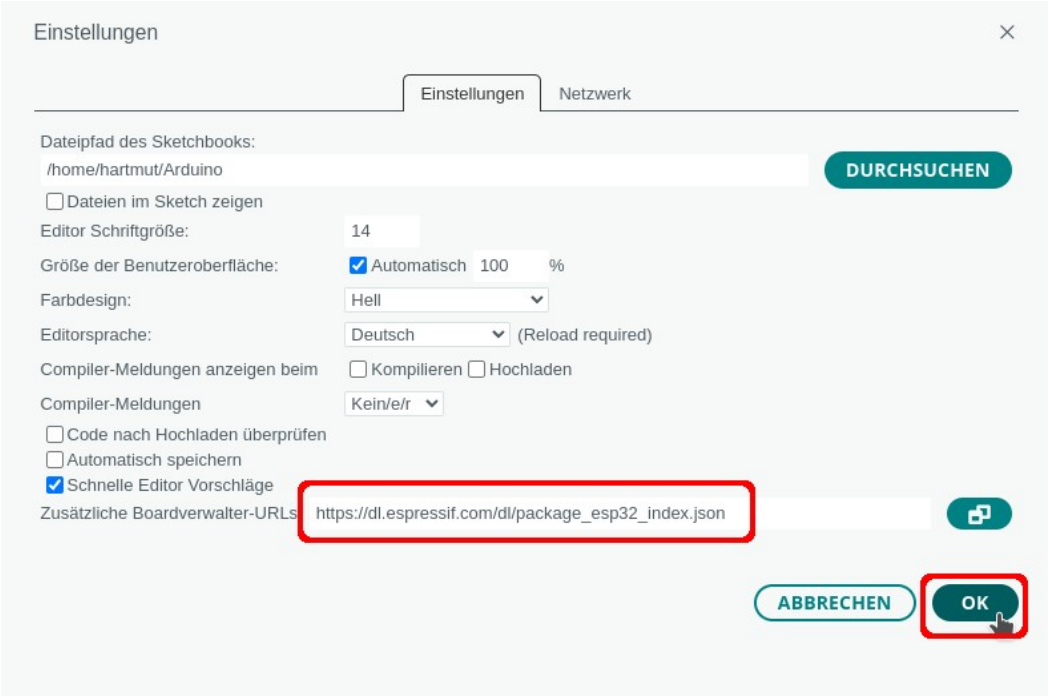
Pinbelegung Arduino Nano ESP32



ESP32-Wroom: Board installieren:

Trage unter Datei -> Einstellungen eine zusätzliche Boardverwalter-URL ein:

https://espressif.github.io/arduino-esp32/package_esp32_dev_index.json



➔ Icon für den Boardverwalter anklicken oder:

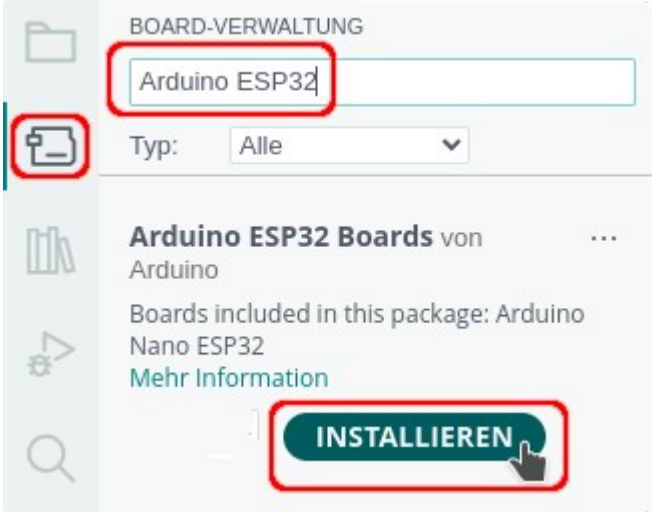
➔ Werkzeuge-> Board -> Boardverwalter

➔ nach ESP32 suchen

➔ Board installieren

Anschließend wird das Board ausgewählt:

Arduino Nano ESP32: Board installieren:



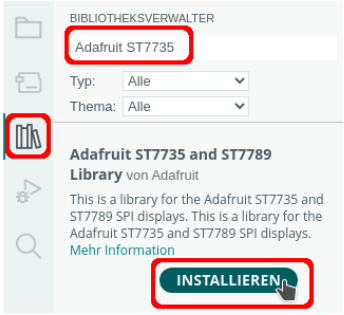
➔ Icon für den Boardverwalter anklicken oder:

➔ Werkzeuge-> Board -> Boardverwalter

➔ nach ESP32 suchen

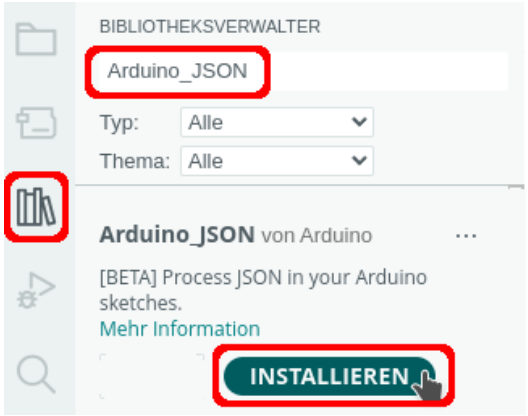
➔ **Board installieren**

Benötigte Bibliotheken



1,77 Zoll/1,8 Zoll TFT

2,4 Zoll TFT

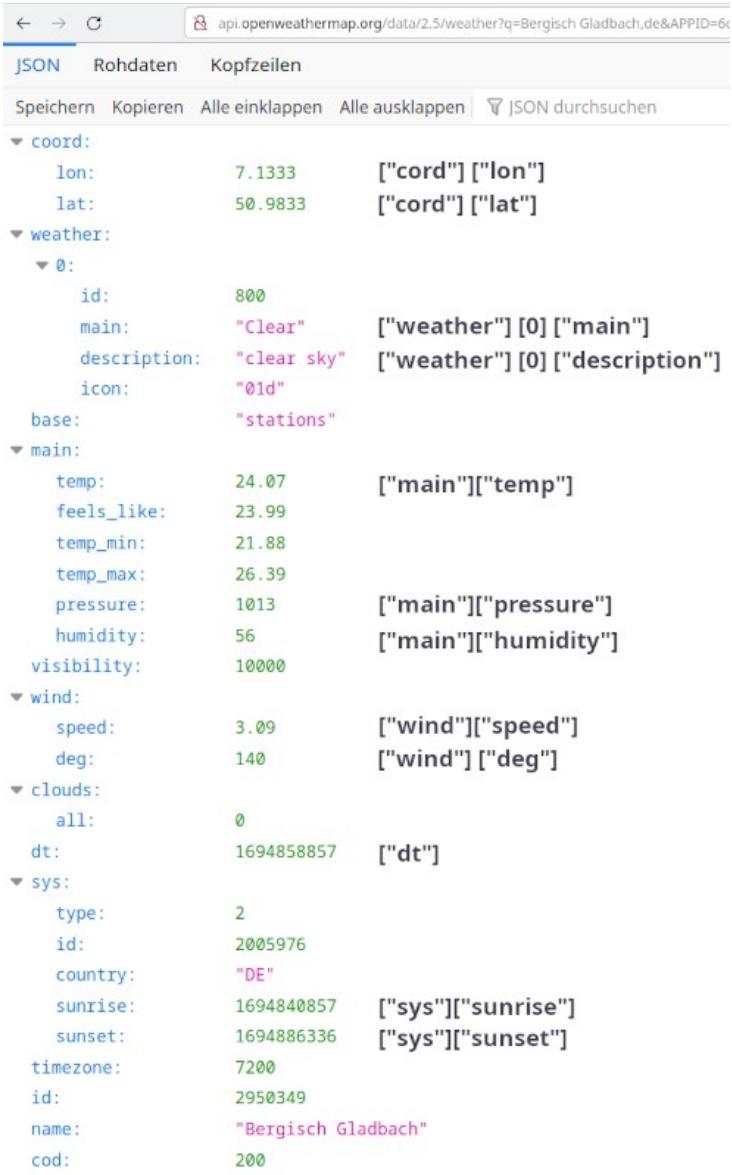


Erläuterung zu JSON

Die Daten liegen im JSON-Format (JavaScript Object Notation) vor. JSON dient dem Austausch von Daten zwischen einem Server und einer Webanwendung. JSON-Daten sind eine Sammlung von Schlüssel-Wert-Paaren. Die Bibliothek filtert aus den Daten diese Schlüssel-Wert-Paare heraus.

Beispiel JSON-Werte beim Aufruf für Bergisch Gladbach

Schlüssel und Wert werden in eckige Klammern eingeschlossen.



Abruf der Daten von openweathermap.org

http://api.openweathermap.org/data/2.5/weather?q=Bergisch %20Gladbach,de&APPID=6d320ceb0961bbfc928aa313xxxxxxxx&units=metric

q= → Name der Stadt, Länderkürzel
APPID → deine APPID (ich habe meine eigene unkenntlich gemacht)
units=metric → metrische Maßangaben, die Temperatur wird als Standard in Kelvin angezeigt

Das Programm

Einbinden der Bibliotheken und Definition der Variablen

Der einzige Unterschied zwischen den beiden Microcontrolern ist die Zuordnung der SPI-Pins.

```
#include "WiFi.h"
#include "HTTPClient.h"
#include "Arduino_JSON.h"
#include "time.h"

# include "Adafruit_GFX.h"
# include "Adafruit_ST7735.h"

/*
 SPI-Pins ESP32-Wroom
  DIN 23
  CLK 18
  CS 3
  RST 22
  DC 2
*/
# define TFT_CS        5
# define TFT_RST      22
# define TFT_DC        2

/*
  SPI-Pins Arduino Nano ESP32
  DIN 11
  CLK 13
  CS 10
  RST 8
  DC 9
  # define TFT_CS 10
  # define TFT_RST 8
  # define TFT_DC  9
*/
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

char Router[] = "Router_SSID";
char Passwort[] = "xxxxxxxx";

// NTP-Server aus dem Pool
# define Zeitserver "de.pool.ntp.org"

/*
  Liste der Zeitzonen
  https://github.com/nayarsystems/posix_tz_db/blob/master/zones.csv
  Zeitzone CET = Central European Time -1 -> 1 Stunde zurück
  CEST = Central European Summer Time von
  M3 = März, 5.0 = Sonntag 5. Woche, 02 = 2 Uhr
  bis M10 = Oktober, 5.0 = Sonntag 5. Woche 03 = 3 Uhr
*/
```

```
#define Zeitzone "CET-1CEST,M3.5.0/02,M10.5.0/03"

// time_t enthält die Anzahl der Sekunden seit dem 1.1.1970 0 Uhr
time_t aktuelleZeit;

/*
  Struktur tm
  tm_hour -> Stunde: 0 bis 23
  tm_min -> Minuten: 0 bis 59
  tm_sec -> Sekunden 0 bis 59
  tm_mday -> Tag 1 bis 31
  tm_mon -> Monat: 0 (Januar) bis 11 (Dezember)
  tm_year -> Jahre seit 1900
  tm_yday -> vergangene Tage seit 1. Januar des Jahres
  tm_isdst -> Wert > 0 = Sommerzeit (dst = daylight saving time)
*/
tm Zeit;

// Daten dür die API von Openweather
String APIKey = "6d320ceb0961bbfc928aa313a7bc9979";
String Stadt = "Bergisch Gladbach";
String Land = "DE";

// Aktualisierungs-Intervall
unsigned long Intervall = 60000;

// String für die vom Server gelieferten Daten
String JSONDaten;
```

setup-Teil

```
void setup()
{
  // Zeitzone: Parameter für die zu ermittelnde Zeit
  configTzTime(Zeitzone, Zeitserver);

  Serial.begin(9600);

  // WiFi starten und Verbindung aufbauen
  WiFi.begin(Router, Passwort);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(200);
    Serial.print(".");
  }

  // SSID des Routers anzeigen
  Serial.println();
  Serial.print("Verbunden mit ");
  Serial.println(WiFi.SSID());

  // IP anzeigen
  Serial.print("IP: ");
  Serial.println(WiFi.localIP());
```

```
  // TFT starten schwarzer Hintergrund
  tft.initR(INITR_BLACKTAB);

  // Rotation anpassen Querformat
  tft.setRotation(1);

  // Schriftgröße
  tft.setTextSize(1);
}
```

Funktion ServerAntwortHolen()

Im loop-Teil wird die Funktion ServerAntwortholen() aufgerufen. Sie holt die Wetterdaten als String, der im loop-Teil in Schlüssel-Wert-Paare umgewandelt wird.

```
String ServerAntwortHolen(const char* OpenweatherServer)
{
  WiFiClient Client;
  HTTPClient httpClient;

  httpClient.begin(Client, OpenweatherServer);

  // Anfrage senden
  int AntwortCode = httpClient.GET();

  String ServerAntwort = "";

  if (AntwortCode > 0)
  {
    // Serial.print("Antwort-Code: ");
    // Serial.println(AntwortCode);

    // Wetter als String holen, wird später in ein JSON-Objekt umgewandelt
    ServerAntwort = httpClient.getString();
  }

  else {
    Serial.print("Error code: ");
    Serial.println(ServerAntwort);
  }

  httpClient.end();

  return ServerAntwort;
}
```

loop-Teil

```
void loop()
{
    tft.fillScreen(ST7735_BLACK);
    tft.setTextColor(ST7735_GREEN);

    tft.setCursor(1, 5);

    // aktuelle Zeit holen
    time(&aktuelleZeit);

    // localtime_r -> Zeit in die lokale Zeitzone setzen
    localtime_r(&aktuelleZeit, &Zeit);

    // Tag: führende 0 ergänzen
    if (Zeit.tm_mday < 10)
    {
      Serial.print("0");
      tft.print("0");
    }
    Serial.print(Zeit.tm_mday);
    Serial.print(".");
    tft.print(Zeit.tm_mday);
    tft.print(".");

    // Monat: führende 0 ergänzen
    if (Zeit.tm_mon < 10)
    {
      Serial.print("0");
      tft.print("0");
    }

    // Zählung des Monats beginnt mit 0 -> 1 hinzufügen
    Serial.print(Zeit.tm_mon + 1);
    Serial.print(".");
    tft.print(Zeit.tm_mon + 1);
    tft.print(".");

    // Anzahl Jahre seit 1900
    Serial.print(Zeit.tm_year + 1900);
    Serial.print(" ");
    tft.print(Zeit.tm_year + 1900);
    tft.print(" ");

    // Stunde: wenn Stunde < 10 -> 0 davor setzen
    if (Zeit.tm_hour < 10)
    {
      Serial.print("0");
      tft.print("0");
    }
    Serial.print(Zeit.tm_hour);
    Serial.print(":");
    tft.print(Zeit.tm_hour);
    tft.print(":");
```

```
  // Minuten
  // wenn Minute < 10 -> 0 davor setzen
  if (Zeit.tm_min < 10)
  {
    Serial.print("0");
    tft.print("0");
  }
  Serial.print(Zeit.tm_min);
  Serial.print(":");
  tft.print(Zeit.tm_min);
  tft.print(":");

  // Sekunden
  if (Zeit.tm_sec < 10)
  {
    Serial.print("0");
    tft.print("0");
  }
  Serial.print(Zeit.tm_sec);
  Serial.println();
  tft.print(Zeit.tm_sec);

  // Wetterdaten holen, wenn WiFi verbunden ist
  if (WiFi.status() == WL_CONNECTED)
  {
    // Name des Servers und Daten übergeben
    String OpenweatherServer = "http://api.openweathermap.org/data/2.5/weather?q=" + Stadt + ",";
    OpenweatherServer = OpenweatherServer + Land + "&APPID=" + APIKey + "&units=metric";

    // Daten vom Server abrufen
    // c_str() liefert einen mit \0 beendeten String
    JSONDaten = ServerAntwortHolen(OpenweatherServer.c_str());

    // wenn die Stadt nicht gefunden wurde
    if (JSONDaten.indexOf("city not found") > 0)
    {
      Serial.println("Stadt nicht gefunden!");
      tft.setCursor(1, 40);
      tft.println("Stadt nicht gefunden!");
      while(1);
    }

    /*
      parse: Zeichenkette im JSON-Format in ein JavaScript-Objekt umzuwandeln
      damit die Daten (Schlüssel-Wert-Paare)ausgewertet werden können
      z.B. ["main"] ["temp"]
    */
    JSONVar Objekt = JSON.parse(JSONDaten);

    // Stadt
    Serial.println(Stadt);
    tft.setCursor(1,15);
    tft.println(Stadt);
    tft.drawFastHLine(1, 25, tft.width(), ST7735_GREEN);
    tft.setTextColor(ST7735_WHITE);
```

```
// Temperatur
Serial.print("Temperatur: ");
double Temperatur = Objekt["main"]["temp"];
String AnzeigeTemperatur = String(Temperatur);
AnzeigeTemperatur.replace(".", ",");
Serial.print(AnzeigeTemperatur);
Serial.println("°C");
tft.setCursor(1,33);
tft.print("Temperatur: " + AnzeigeTemperatur + char(247) + "C");

// Luftdruck
Serial.print("Luftdruck: ");
Serial.print(Objekt["main"]["pressure"]);
Serial.println(" hPa");
tft.setCursor(1,46);
tft.print("Luftdruck: ");
tft.print(Objekt["main"]["pressure"]);
tft.println(" hPa");

// Luftfeuchtigkeit
Serial.print("Luftfeuchtigkeit: ");
Serial.print(Objekt["main"]["humidity"]);
Serial.println("%");
tft.setCursor(1,59);
tft.print("Luftfeuchtigkeit: ");
tft.print(Objekt["main"]["humidity"]);
tft.println("%");

// Windgeschwindigkeit
Serial.print("Windgeschwindigkeit: ");
double Windgeschwindigkeit = Objekt["wind"]["speed"];
String AnzeigeWindgeschwindigkeit = String(Windgeschwindigkeit);
AnzeigeWindgeschwindigkeit.replace(".", ",");
Serial.print(AnzeigeWindgeschwindigkeit);
Serial.println(" m/s");
tft.setCursor(1,72);

tft.print("Wind: " + AnzeigeWindgeschwindigkeit);
tft.println(" m/s");

// Windrichtung
Serial.print("Windrichtung: ");
Serial.print(Objekt["wind"]["deg"]);
Serial.println("°");

// Bewölkung
Serial.print("Wetterlage: ");
String Wetterlage = Objekt["weather"][0]["main"];
tft.setCursor(1,85);

if (Wetterlage == "Clear")
{
  Serial.println("klarer Himmel");
  tft.print("klarer Himmel");
}
```

```
if (Wetterlage == "Mist")
{
  Serial.println("Nebel");
  tft.print("Nebel");
}

if (Wetterlage == "Clouds")
{
  Serial.println("wolkig");
  tft.println("wolkig");
}

if (Wetterlage == "Rain")
{
  Serial.println("Regen");
  tft.println("Regen");
}

if (Wetterlage == "Snow")
{
  Serial.println("Schneefall");
  tft.println("Schneefall");
}

// Serial.println(Objekt["weather"][0]["description"]);

// Sonnenaufgang
time_t Sonnenaufgang;
time(&Sonnenaufgang);
Sonnenaufgang = Objekt["sys"]["sunrise"];
Serial.print("Sonnenaufgang: ");

// ctime: Unix-Zeit in "lesbares" Format umwandeln
String ZeitSonnenaufgang = ctime(&Sonnenaufgang);

tft.setCursor(1, 98);

// Uhrzeit extrahieren
Serial.println(ZeitSonnenaufgang.substring(11, 16));

// Sonnenuntergang
time_t Sonnenuntergang;
time(&Sonnenuntergang);
Sonnenuntergang = Objekt["sys"]["sunset"];
Serial.print("Sonnenuntergang: ");
String ZeitSonnenuntergang = ctime(&Sonnenuntergang);
Serial.println(ZeitSonnenuntergang.substring(11, 16));
tft.print("Sonnenuntergang: ");

tft.println(ZeitSonnenuntergang.substring(11, 16));

// letzte Messung der Wetterdaten
time_t letzteMessung;
time(&letzteMessung);
letzteMessung = Objekt["dt"];
Serial.print("Letzte Messung: ");
```

```
      String ZeitMessung = ctime(&letzteMessung);
      Serial.println(ZeitMessung.substring(11, 16));
      Serial.println("--------------------------");
      tft.setCursor(1, 110);
      tft.print("Letzte Messung: " + ZeitMessung.substring(11, 16));
    }

    delay(Intervall);
}
```

Hartmut Waller (https://hartmut-waller.info/arduinoblog) letzte Änderung: 12.11.23