

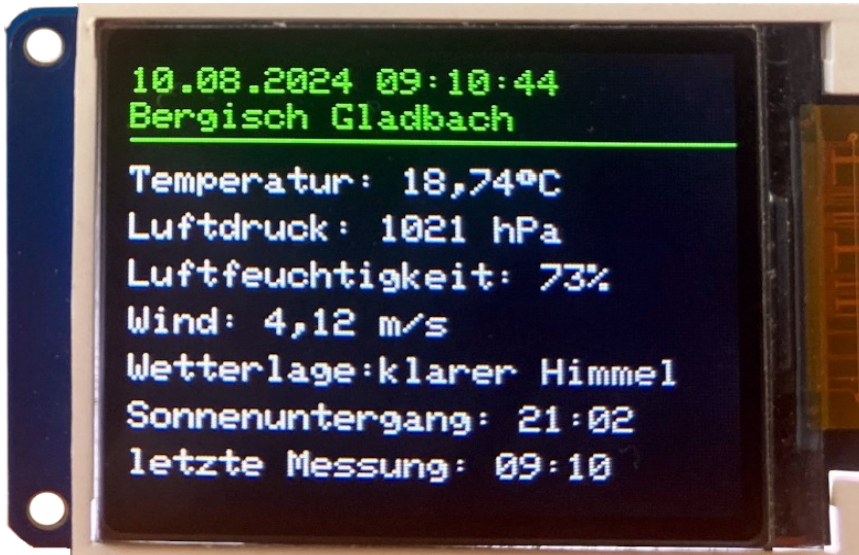
Inhaltsverzeichnis

Darstellung der Daten.....	1
TFT-Display.....	1
Serieller Monitor.....	1
Vorbereitung.....	2
Benötigte Bauteile.....	2
Pinbelegung ESP32-Wroom.....	2
Pinbelegung Arduino Nano ESP32.....	3
ESP32-Wroom: Board installieren:.....	3
Arduino Nano ESP32: Board installieren:.....	4
Benötigte Bibliotheken.....	4
Erläuterung zu JSON.....	5
Abruf der Daten von openweathermap.org.....	6
Vorbereitung.....	6
Aufruf der API.....	7
Das Programm.....	7
Einbinden der Bibliotheken und Definition der Variablen.....	7
setup-Teil.....	9
Funktion ServerAntwortHolen().....	9
loop-Teil.....	10

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://openweathermap.org/api>



Um den Zugang zu nutzen, musst du deine Zahlungsdaten hinterlegen.

Der API-Schlüssel erlaubt 1000 Zugriffe am Tag. Das entspricht einem Zugriff alle 86,4 Sekunden.

Quelle: <https://openweathermap.org/price#weather>

Um sicherzugehen kannst du die Anzahl der Zugriffe beschränken:

Quelle: <https://home.openweathermap.org/subscriptions>

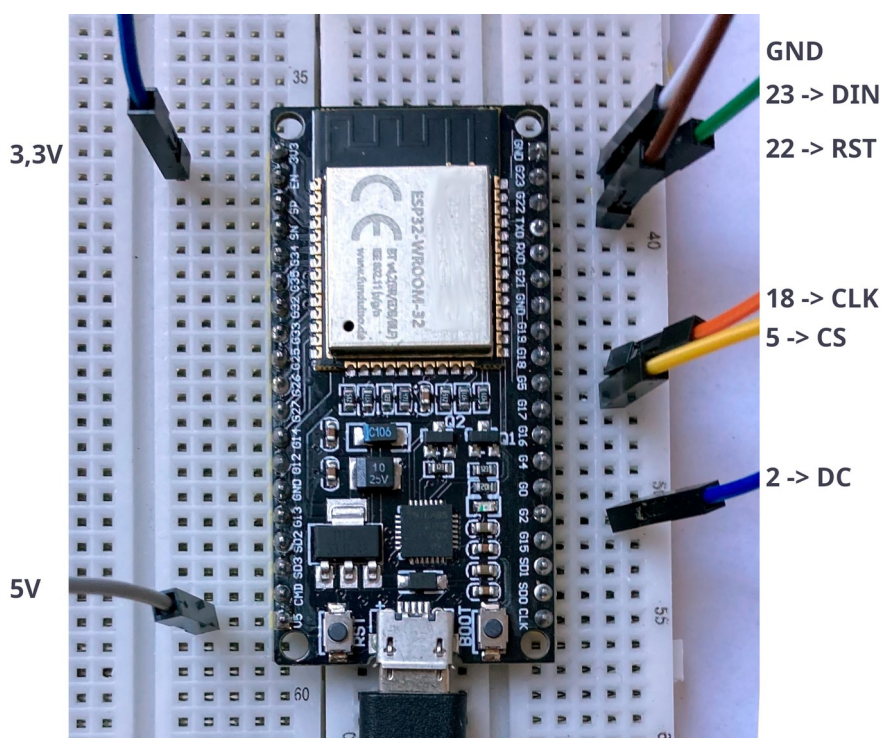
Du kannst die Anzahl der Zugriffe feststellen:

Quelle: https://home.openweathermap.org/statistics/onecall_30

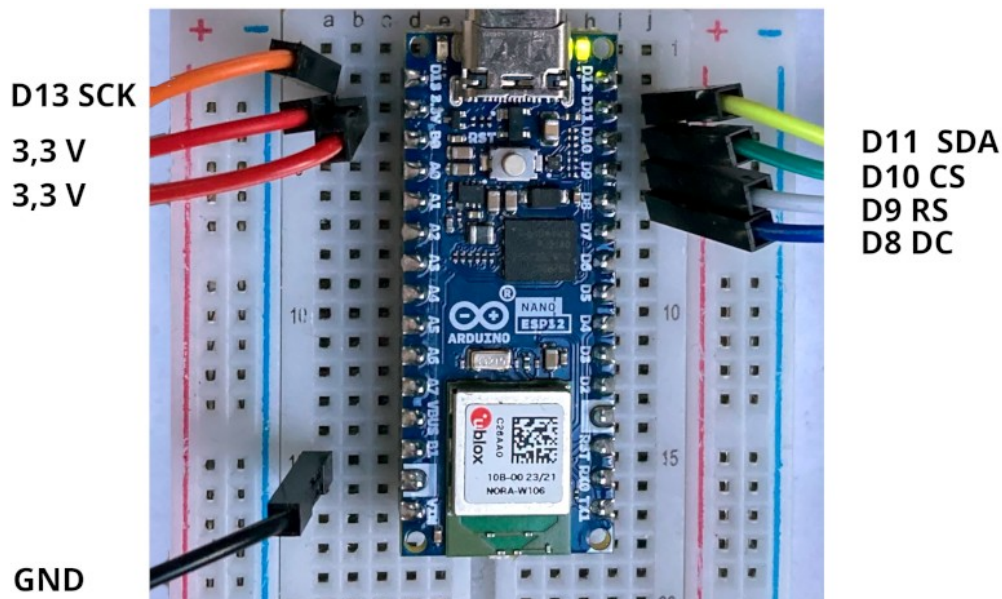
Benötigte Bauteile

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

Pinbelegung ESP32-Wroom



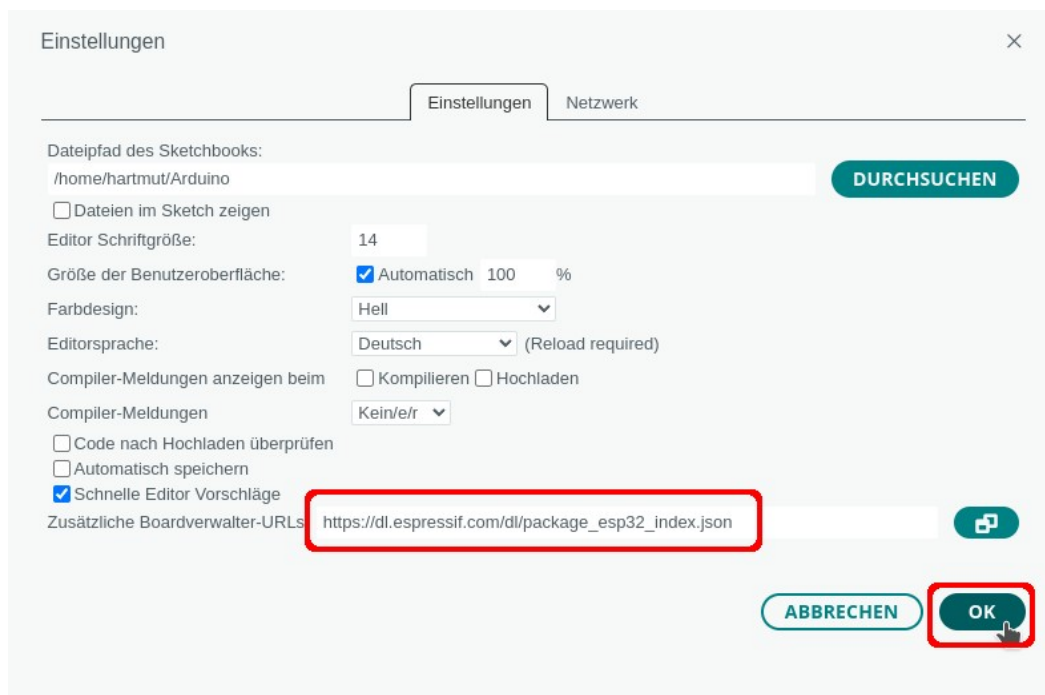
Pinbelegung Arduino Nano ESP32

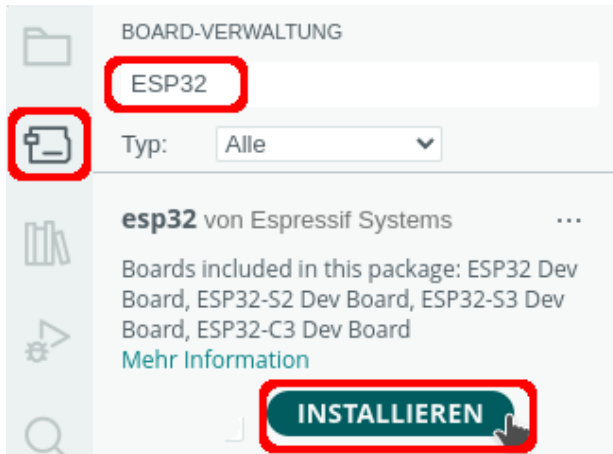


ESP32-Wroom: Board installieren:

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

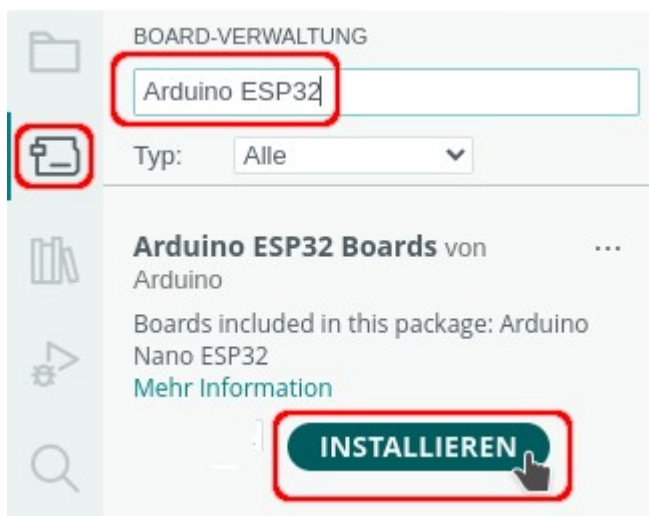
→ nach ESP32 suchen

→ Board installieren



Anschließend wird das Board ausgewählt:

Arduino Nano ESP32: Board installieren:



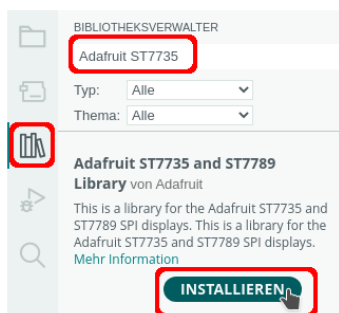
→ Icon für den Boardverwalter anklicken
oder:

→ Werkzeuge-> Board ->
Boardverwalter

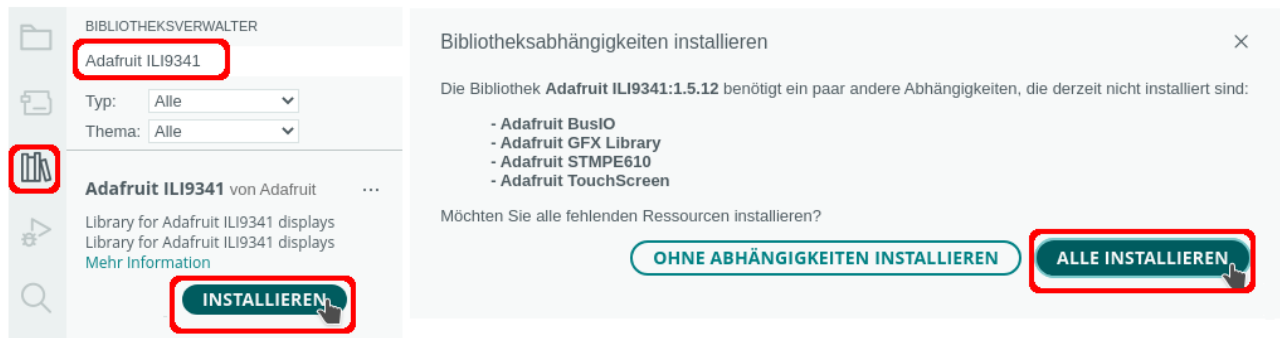
→ nach ESP32 suchen

→ **Board installieren**

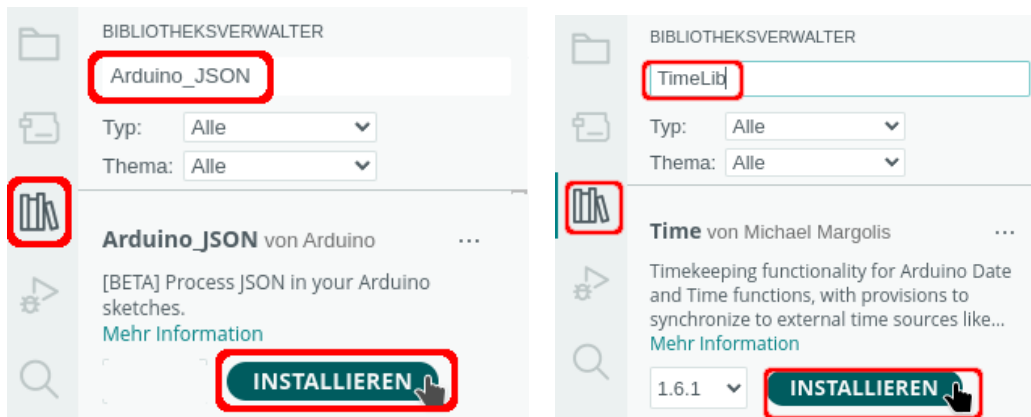
Benötigte Bibliotheken



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.


```
lat: 50.9911 ["lat"]
lon: 7.1291 ["lon"]
timezone: "Europe/Berlin" ["timezone"]
timezone_offset: 7200

current:
  dt: 1723232227 ["current"]["dt"]
  sunrise: 1723176583 ["current"]["sunrise"]
  sunset: 1723230257 ["current"]["sunset"]
  temp: 21.15 ["current"]["temp"]
  feels_like: 21.59
  pressure: 1018 ["current"]["pressure"]
  humidity: 87 ["current"]["humidity"]
  dew_point: 18.9
  uvi: 0
  clouds: 100
  visibility: 10000
  wind_speed: 2.68 ["current"]["wind_speed"]
  wind_deg: 250 ["current"]["wind_deg"]
  wind_gust: 8.31

weather:
  0:
    id: 804
    main: "Clouds" ["current"]["weather"][0]["main"]
    description: "overcast clouds" ["current"]["weather"][0]["description"]
    icon: "04n"
```

Abruf der Daten von openweathermap.org

Vorbereitung

Du benötigst die Geokoordinaten (Längengrad = lon, Breitengrad = lat) des gewünschten Orts. Am einfachsten geht das mit Openweathermap selbst.

Weather in your city

Search

Bergisch Gladbach, DE *clear sky*

17.5°C temperature from 15.5 to 18.2 °C, wind 1.34 m/s. clouds 0 %, 1021 hpa

Geo coords [50.9833, 7.1333]

Aufruf der API

http://api.openweathermap.org/data/3.0//onecall?
&lat=50.99111161485325&lon=7.129065378199176&APPID=xxxxxxx&units=metric&exclude=daily,hourly,minutely

- ➔ lat, lon -> Breitengrad, Längengrad
- ➔ APPID -> deine APPID (ich habe meine eigene unkenntlich gemacht)
- ➔ units=metric -> metrische Maßangaben, die Temperatur wird als Standard in Kelvin angezeigt
- ➔ exclude=daily,hourly,minutely -> keine Daten der Wettervorhersage anzeigen
- ➔ lang=de -> Ausgabe der Beschreibungen auf deutsch



Manchmal werden mehrere Versuche benötigt um die Zeit zu synchronisieren und den Openweather-Server zu erreichen.

Das Programm

Einbinden der Bibliotheken und Definition der Variablen

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

```
#include "WiFi.h"
#include "HTTPClient.h"
#include "Arduino_JSON.h"
#include "time.h"
#include "TimeLib.h"
#include "Adafruit_GFX.h"
#include "Adafruit_ST7735.h"

/*
  SPI-Pins
  DIN 23
  CLK 18
  CS 5
  RST 22
  DC 2
*/
# define TFT_CS      5
# define TFT_RST     22
# define TFT_DC      2
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

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

// 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 für die API von Openweather -> muss angepasst werden
String APIKey = "4d26a936a4fe519ff3678dxxxxxxxxxx";

// lon = longitude = Breitengrad, lat = latitude = Längengrad
// mit Kartenprogramm oder Openweathermap bestimmen
// Daten für Bergisch Gladbach
String Koordinaten = "&lat=50.99111161485325&lon=7.129065378199176";
String Stadt = "Bergisch Gladbach";

/*
  Aktualisierungs-Intervall
  1000 Zugriffe pro Tag kostenlos
  86.400 Sekunden = 1 Tag -> Abruf alle 86,4 Sekunden möglich
  zu Testzwecken das Intervall kurz halten und später
  auf 10 Minuten (oder länger) zu setzen
*/
unsigned long Intervall = 600000;

// String für die vom Server gelieferten Rohdaten
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 = "";
```

```
// Antwort erhalten: Code 200
if (AntwortCode == 200)
{
    // Wetter als String holen, wird später in ein JSON-Objekt umgewandelt
    ServerAntwort = httpClient.getString();

    // Rohdaten anzeigen
    // Serial.println(Serverantwort);
}

else
{
    Serial.println("Der Server ist nicht erreichbar!");
}

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 < 9)
    {
        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");
}
Serial.print(Zeit.tm_hour);
Serial.print(":");

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

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

// Wetterdaten holen, wenn WiFi verbunden ist
if (WiFi.status() == WL_CONNECTED)
{
    // Name des Servers und Daten übergeben
    String OpenweatherServer = "http://api.openweathermap.org/data/3.0//onecall?" +
Koordinaten;
    OpenweatherServer = OpenweatherServer + "&APPID=" + APIKey +
"&units=metric&exclude=daily,hourly,minutely";

    // Server anzeigen
    Serial.println(OpenweatherServer);

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

    /*
        parse: Zeichenkette im JSON-Format in ein JavaScript-Objekt umzuwandeln
        damit die Daten (Schlüssel-Wert-Paare) ausgewertet werden können
        z.B. ["current"] ["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["current"]["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["current"]["pressure"]);
Serial.println(" hPa");
tft.setCursor(1, 46);
tft.print("Luftdruck: ");
tft.print(Objekt["current"]["pressure"]);
tft.println(" hPa");

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

// Windgeschwindigkeit
Serial.print("Windgeschwindigkeit: ");
double Windgeschwindigkeit = Objekt["current"]["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["current"]["wind_deg"]);
Serial.println("°");

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

```
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");
}

if (Wetterlage == "Drizzle")
{
    Serial.println("Nieselregen");
    tft.println("Nieselregen");
}

if (Wetterlage == "Thunderstorm")
{
    Serial.println("Gewitter");
    tft.println("Gewitter");
}

// letzte Messung entspricht der gerade ermittelten Zeit
Serial.print("letzte Messung: ");

tft.setCursor(1, 180);
tft.print("letzte Messung: ");

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

Serial.println(Zeit.tm_min);
tft.print(Zeit.tm_min);

// Sonnenaufgang als UNIX-Time
long Sonnenaufgang = Objekt["current"]["sunrise"];
Serial.print("Sonnenaufgang: ");

// Zeit des Sonnenaufgangs setzen
setTime(Sonnenaufgang);
String ZeitSonnenaufgang;

// Uhrzeit bestimmen
// Sommerzeit
if(Zeit.tm_isdst > 0)
{
    if (hour(Sonnenaufgang) + 2 < 10) ZeitSonnenaufgang = "0";
    ZeitSonnenaufgang = ZeitSonnenaufgang + String(hour(Sonnenaufgang) + 2) + ":";
}

// Normalzeit
else
{
    if (hour(Sonnenaufgang) + 1 < 10) ZeitSonnenaufgang = "0";
    ZeitSonnenaufgang = ZeitSonnenaufgang + String(hour(Sonnenaufgang) + 1) + ":";
}

if (minute(Sonnenaufgang) < 10) ZeitSonnenaufgang = ZeitSonnenaufgang + "0";
ZeitSonnenaufgang = ZeitSonnenaufgang + String(minute(Sonnenaufgang));
Serial.println(ZeitSonnenaufgang);

// Sonnenuntergang
long Sonnenuntergang = Objekt["current"]["sunset"];
Serial.print("Sonnenuntergang: ");
setTime(Sonnenuntergang);
String ZeitSonnenuntergang;

// Sommerzeit
if(Zeit.tm_isdst > 0)
{
    if (hour(Sonnenuntergang) + 2 < 10) ZeitSonnenuntergang = "0";
    ZeitSonnenuntergang = ZeitSonnenuntergang + String(hour(Sonnenuntergang) + 2) + ":";
}

// Normalzeit

```

```
else
{
    if (hour(Sonnenuntergang) + 1 < 10) ZeitSonnenuntergang = "0";
    ZeitSonnenuntergang = ZeitSonnenuntergang + String(hour(Sonnenuntergang) + 1) + ":";
}

if (minute(Sonnenuntergang) < 10) ZeitSonnenuntergang = ZeitSonnenuntergang + "0";
ZeitSonnenuntergang = ZeitSonnenuntergang + String(minute(Sonnenuntergang));
Serial.println(ZeitSonnenuntergang);
Serial.println("-----");

tft.setCursor(1, 111);

// zwischen 0 und 11 Uhr -> Sonnenaufgang anzeigen
// zwischen 12 und 23 Uhr Sonnenuntergang anzeigen
switch (Zeit.tm_hour)
{
    case 0 ... 11:
        tft.print("Sonnenaufgang: " + ZeitSonnenaufgang);
        break;
    case 12 ... 23:
        tft.print("Sonnenuntergang: " + ZeitSonnenuntergang);
        break;
}
}

delay(Intervall);
}
```