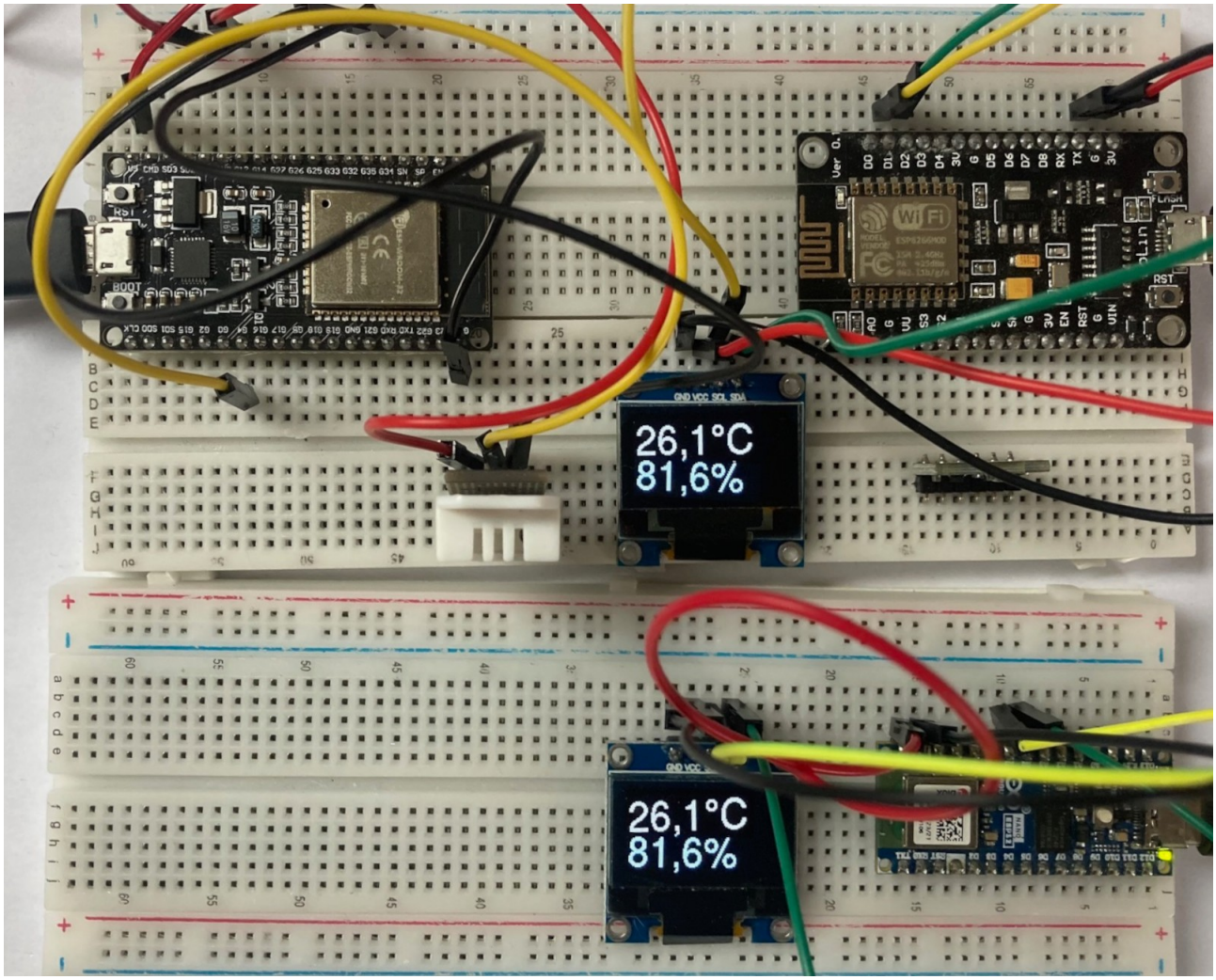
Inhaltsverzeichnis

Ziel des Projekts

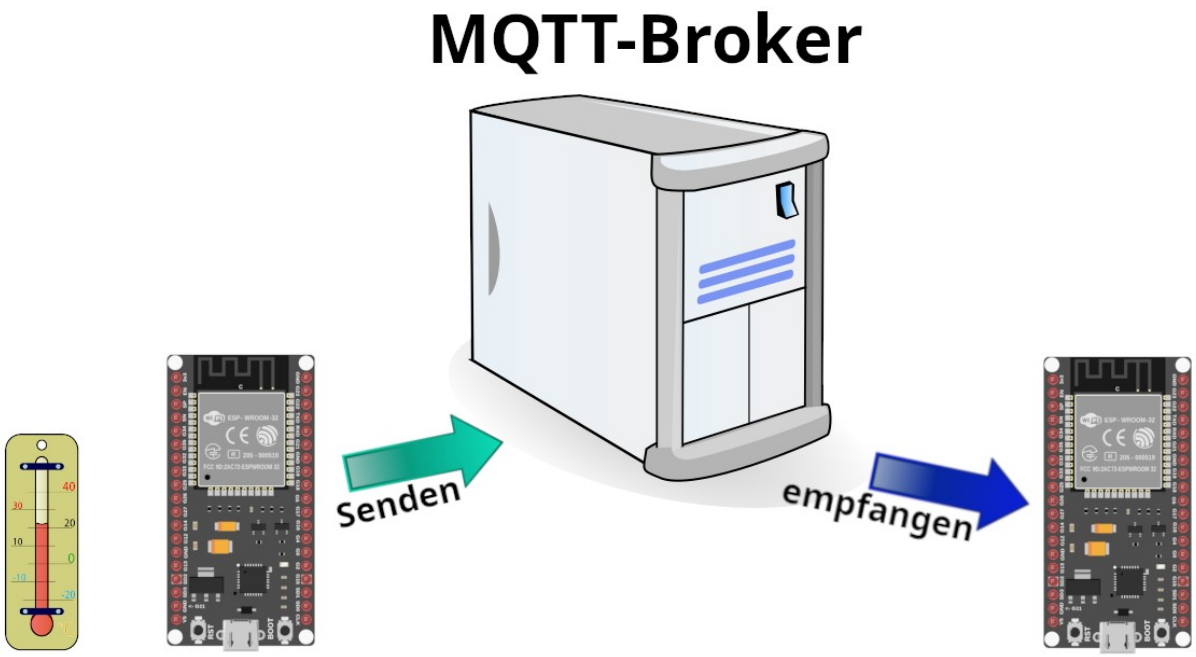
Ein ESP32-Mikrocontroller misst die Temperatur. Die Daten werden an einen MQTT-Server gesendet, ein zweiter ESP32 empfängt die Daten und zeigt sie auf einem OLED an.



ESP32-Wroom: Messung NodeMCU/Arduino Nano ESP32 Anzeige auf OLED

MQTT

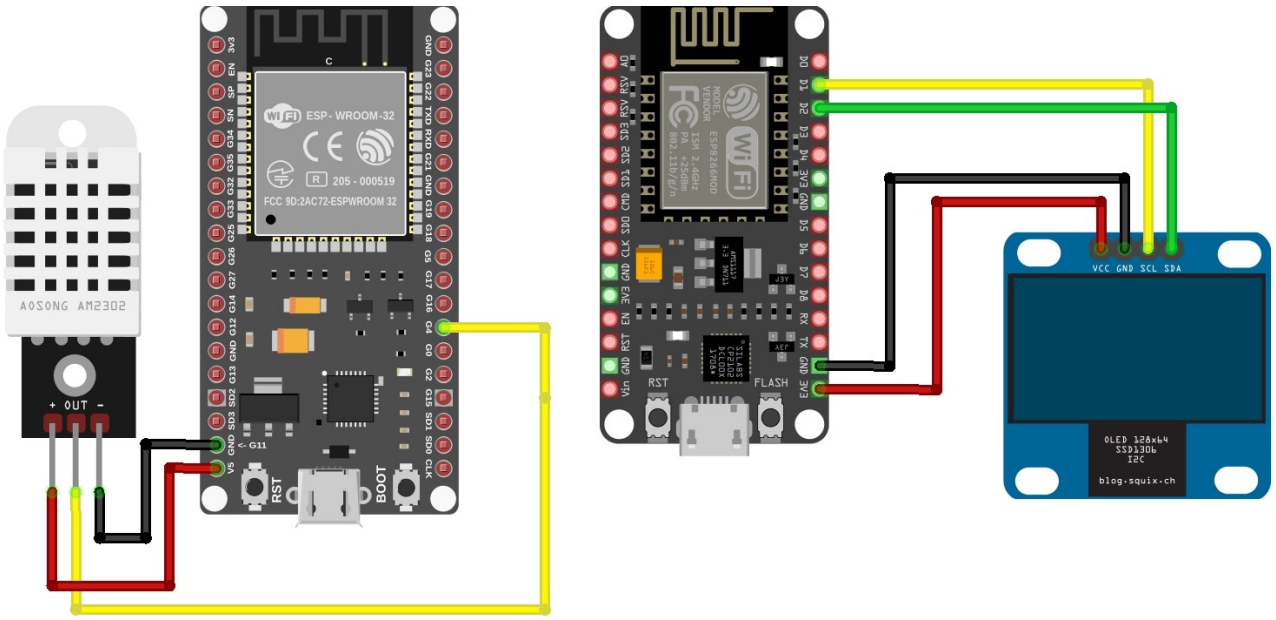
MQTT (Message Queue Telemetry Transport) ist ein Protokoll für die Übertragung von Daten nach dem Publisher-Subscriber-Modell. Es besteht aus dem Server (Broker) und verschiedenen Klienten. Einer dieser Klienten veröffentlicht (Publish) zu einem Thema (Topic) eine Nachricht und schickt sie an den Broker. Ein anderer Klient sendet eine Anfrage (Subscribe) zum Broker und fragt nach Daten (Payload) zum festgelegten Thema. Sind Daten vorhanden, werden sie übermittelt.



Benötigte Bauteile

- ESP32-Wroom/NodeMCU/Arduino Nano ESP32 als Messgerät
- ESP32-Wroom/NodeMCU/Arduino Nano ESP32 für die Anzeige
- Leitungsdrähte
- OLED
- DHT22

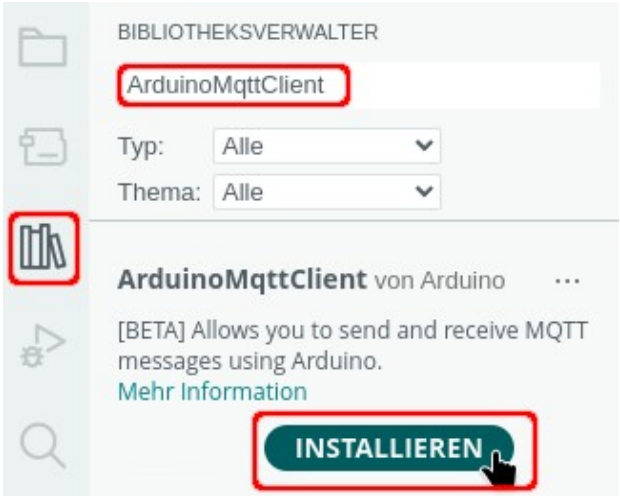
Der Schaltplan

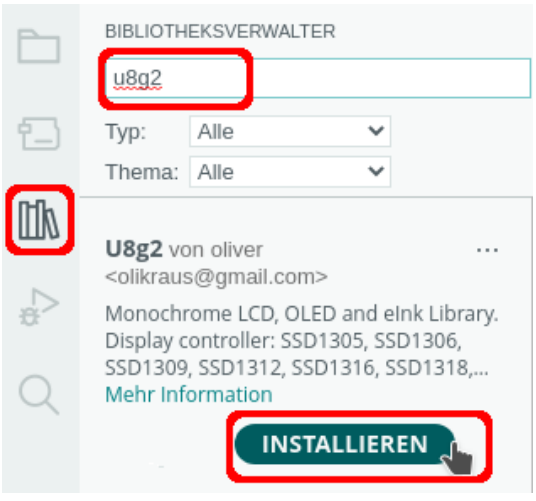


Sender (Publisher) **Empfänger (Subscriber)**

Das Programm

Benötigte Bibliotheken





Funktionen Bibliothek u8g2

Schlüsselwort	Parameter	Aktion
begin();		OLED-Display starten
getDisplayWidth();		Bildschirmbreite feststellen
getDisplayHeight();		Bildschirmhöhe feststellen
setdrawColor(Parameter)	0 → schwarz 1 → weiß	Zeichenfarbe festlegen
clearDisplay();		Bildschirm dunkel schalten
setContrast(Parameter)	0 … 255	Kontrast einstellen
setDisplayRotation(U8G2_*);	U8G2_0 → 0 Grad U8G2_1 → 90 Grad U8G2_2 → 180 Grad U8G2_3 → 270 Grad	Anzeige drehen
flipMode(Parameter);	0 → normale Ausrichtung 1 → 180 Grad drehen wirksam erst bei einer Rotation	Anzeige spiegeln (180 Grad)
home();		Cursor in die linke obere Ecke setzen
drawPixel(x-Achse, y-Achse)		einzelnen Pixel zeichnen
drawLine(StartX, StartX, EndeX, EndeY);		Linie zeichnen
drawHLine(StartX, StartY, Länge);		horizontale Linie zeichnen
drawVLine(StartX, StartY, Länge);		vertikale Linie zeichnen
drawFrame(StartX, StartY,, Breite, Höhe);		Rechteck zeichnen
drawRFrame(StartX, StartY, Breite, Höhe, Eckenradius);		abgerundetes Rechteck zeichnen
drawBox(StartX, StartY, Breite, Höhe);		ausgefülltes Rechteck zeichnen
drawRBox(StartX, StartY, Breite, Höhe, Eckenradius);		abgerundetes ausgefülltes Rechteck zeichnen
drawCircle(MittelpunkX, MittelpunktY, Radius);		Kreis zeichnen
drawDisc(MittelpunkX, MittelpunktY, Radius);		ausgefüllten Kreis zeichnen
drawXBM(StartX, StartY, Breite, Höhe, Array_Bilddatei);		XBM-Bild anzeigen
drawEllipse(StartX, StartY, RadiusX, RadiusY);		Ellipse zeichnen
print("Text"); drawStr(StartX, StartY ,"Text");		Text schreiben
setFontDirection(Wert);	0 → normal ausgerichtet 1 → 90 ° gedreht 2 → 180 ° gedreht 3 → 270 ° gedreht	Schreibrichtung
setCursor(x-Achse, y-Achse);		Cursor setzen

Schlüsselwort	Parameter	Aktion
u8g2.setFont(Schriftart)	**Beispiele für funktionierende Schriftarten:** 6px: u8g2_font_5x7_tr 7px: u8g2_font_torussansbold8_8r 8px: u8g2_font_ncenB08_tr 10px: u8g2_font_t0_15b_me 12px: u8g2_font_helvB12_tf 13px: u8g2_font_t0_22_te 14px: u8g2_font_helvB14_tf 17px: u8g2_font_timB18_tf 18px: u8g2_font_lubB18_tr 20px: u8g2_font_courB24_tf 23px: u8g2_font_timB24_tf 25px: u8g2_font_helvR24_tf 32px: u8g2_font_logisoso32_tf 42px: u8g2_font_fub42_tf 58px: u8g2_font_logisoso58_tf 62px: u8g2_font_logisoso62_tn	Schriftart bestimmen

Funktion der Bibliothek ArduinoMqttClient

Schlüsselwort	Parameter	Aktion
connect();	Name des Brokers, Port des Brokers	Broker verbinden
poll();		Broker abfragen

beginMessage();
print();
endMessage();
	Mindestens Inhalt der Nachricht
	optional:
	Länge der Nachricht,
	auf dem Broker behalten (true/false)
	Qualität der Übermittlung der Nachricht
	0 = Nachricht wird einmal versendet
	eine Überprüfung des Empfangs durch den Broker findet nicht statt
	1 = die Nachricht wird solange versendet, bis der Broker den Empfang bestätigt hat
	2 = die Nachricht wird einmal gesendet
	es wird sicher gestellt, dass die Nachricht beim Broker angekommen ist
	
	Duplikat senden (true/false)
		Nachricht senden starten
		und mit print() übermitteln

beginWill();
print();
endWill();
	Thema
	Länge der Nachricht,
	auf dem Broker behalten (true)
	Qualität der Nachricht (0 .. 2)
		versucht eine getrennte Verbindung neu aufzubauen
		muss vor dem eigentlichen Verbindungsaufbau gestartet werden

onMessage();	Funktion für die Auswertung der empfangen Daten	Daten auswerten
subscribe()	Mindestens das Thema optional: Qualität der Übermittlung der Nachricht	

Öffentliche Broker

Die Verbindungen sind unverschlüsselt, die Erreichbarkeit ist nicht garantiert.

Broker	Port	Benutzer	Passwort
test.mosquitto.org	1883		
public.mqtthq.com	1883		
broker.emqx.io	1883	emqx	public

Der Sender

```
#include "ArduinoMqttClient.h"

// ESP8266: NodeMCU, Wemos D1 Mini
// #include "ESP8266WiFi.h"

// ESP32
#include "WiFi.h"
#include "DHT.h"

// Pin des DHT-Sensors
int SENSOR_DHT = 4;

// DHT11
// #define SensorTyp DHT11

// DHT22
#define SensorTyp DHT22

// dht-Sensor einen Namen und Typ zuweisen
DHT dht(SENSOR_DHT, SensorTyp);

char Router[] = "FRITZ!Box 7590 LB";
char Passwort[] = "anea1246";

WiFiClient wifiClient;

// wifiClient MqttClient zuordnen
MqttClient mqttClient(wifiClient);

const char broker[] = "test.mosquitto.org";
int port = 1883;

/*
  öffentliche Broker
  const char broker[] = "public.mqtthq.com";
  int port = 1883;

  const char broker[] = "test.mosquitto.org";
  int port = 1883;
```

```
  const char broker[] = "broker.emqx.io";
  const char *mqtt_username = "emqx";
  const char *mqtt_password = "public";
  int port = 1883;
*/

// Thema für Senden/Empfangen
const char Messdaten[] = "Messdaten";

// Thema für die "will"-Nachricht für den Versuch die Verbindung
// neu aufzubauen
const char getrennteVerbindung[] = "getrennteVerbindung";
// Intervall zwischen den Messungen
const long Interval = 5000;
unsigned long vorherigeZeit = 0;

void setup()
{
  Serial.begin(9600);
  while (!Serial);
  delay(1000);

  WiFi.begin(Router, Passwort);

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

  Serial.println();
  Serial.print("Verbunden mit ");
  Serial.println(Router);
  Serial.print("IP über DHCP: ");
  Serial.println(WiFi.localIP());

  /*
    will-Nachricht:
    wird an den Broker gesendet und versucht eine getrennte Verbindung neu aufzubauen
    Parameter:
    Thema, Länge des gesendeten Strings, Behalten der Anforderung,
    Qualität der Verbindung
    muss vor dem Aufbau der Verbindung zum Broker festgelegt werden
  */
  String willDaten = "getrennt";
  bool Behalten = true;
  int willQualitaetService= 1;

  // will-Nachricht senden
  mqttClient.beginWill(getrennteVerbindung, willDaten.length(), Behalten, willQualitaetService);
  mqttClient.print(getrennteVerbindung);
  mqttClient.endWill();

  Serial.println("Versuche mit dem Broker " + String(broker) + " zu verbinden ... ");
  delay(1000);
```

```
  // Verbindungsversuch
  if (!mqttClient.connect(broker, port))
  {
    Serial.print("Verbindung zum Broker gescheitert");

    // Programm anhalten
    while (1);
  }
  Serial.println();
  Serial.println("Mit dem Broker " + String(broker) + " verbunden!");
  Serial.println();

  // dht-Sensor starten
  dht.begin();
}

void loop()
{
  // Broker abfragen
  mqttClient.poll();

  unsigned long aktuelleZeit = millis();

  // Intervall zwischen den Sendezeiten
  if (aktuelleZeit - vorherigeZeit >= Interval)
  {
    // letzte Sendezeit sichern
    vorherigeZeit = aktuelleZeit;

    // gemessene Temperatur dht in String umwandeln
    String Temperatur = String(dht.readTemperature(), 1);

    // . durch , ersetzen
    Temperatur.replace(".", ",");

    // gemessene Luftfeuchtigkeit in String umwandeln
    String Luftfeuchtigkeit = String(dht.readHumidity(), 1);

    // . durch , ersetzen
    Luftfeuchtigkeit.replace(".", ",");

    // String Senden zusammenbauen / als Trennzeichen
    String Senden = Temperatur + "/" + Luftfeuchtigkeit;
    Serial.println("Nachricht gesendet:");

    // Anzeige im Seriellen Monitor: String Senden am / trennen
    int Suche = Senden.indexOf("/");

    // String bis zum /
    Temperatur = Senden.substring(0, Suche);

    // String vom ersten Zeichen hinter / bis zum Ende
    Luftfeuchtigkeit = Senden.substring(Suche + 1, Senden.length());
```

```
    // Daten im Seriellen Monitor anzeigen
    Serial.println(Temperatur + "°C");
    Serial.println(Luftfeuchtigkeit + "%");
    Serial.println("---------------------");

    // Messdaten senden
    bool Behalten = false;
    int ServiceQualitaet = 2;
    bool Duplikat = false;
    mqttClient.beginMessage(Messdaten, Senden.length(), Behalten, ServiceQualitaet, Duplikat);
    mqttClient.print(Senden);
    mqttClient.endMessage();
  }
}
```

Der Empfänger

```
#include "ArduinoMqttClient.h"

// ESP32
// #include "WiFi.h"

// ESP8266: NodeMCU, Wemos D1 Mini
#include "ESP8266WiFi.h"

#include "U8g2lib.h"

/*
  Typbezeichnung mit Bildschirmgröße in Pixeln
  1 = page buffer mode, F = full screen buffer mode
  Hardware I2C/Hardware SPI
  Name des OLEDs
  Rotation R0 (keine)
*/
U8G2_SSD1306_128X64_NONAME_1_HW_I2C oled(U8G2_R0, U8X8_PIN_NONE);

// Router SSID/Passwort
char Router[] = "FRITZ!Box 7590 LB";
char Passwort[] = "anea1246";

WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

const char broker[] = "test.mosquitto.org";
int port = 1883;

/* öffentliche Broker
  const char broker[] = "public.mqtthq.com";
  int port = 1883;

  const char broker[] = "test.mosquitto.org";
  int port = 1883;
```

```
  const char broker[] = "broker.emqx.io";
  const char *mqtt_username = "emqx";
  const char *mqtt_password = "public";
  int port = 1883;
*/

// Thema  für Senden/Empfangen
const char Messdaten[] = "Messdaten";

/*
  Qualität der Verbindung (0 .. 2)
  0 = Nachricht wird einmal versendet
      eine Überprüfung des Empfangs durch den Broker findet nicht statt
  1 = die Nachricht wird solange versendet, bis der Broker den Empfang bestätigt hat
  2 = die Nachricht wird einmal gesendet, es wird sicher gestellt, dass sie
      beim Broker angekommen ist
*/
int subscribeQoS = 2;

void setup()
{
  Serial.begin(9600);
  while (!Serial);
  delay(1000);

  WiFi.begin(Router, Passwort);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(200);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Verbunden mit ");
  Serial.println(Router);
  Serial.print("IP über DHCP: ");
  Serial.println(WiFi.localIP());

  Serial.print("Versuche mit dem Broker " + String(broker) + " zu verbinden ... ");

  // Verbindung zum Broker aufbauen
  if (!mqttClient.connect(broker, port))
  {
    Serial.print("Verbindung zum Broker gescheitert");

    // Programm anhalten
    while (1);
  }
  Serial.println();

  Serial.println("Mit dem Broker " + String(broker) + " verbunden!");
  Serial.println();

  // den Rückruf für den Nachrichtenempfang festlegen
  mqttClient.onMessage(empfangeneNachricht);
```

```
  // in das Thema Messdaten "einschreiben"
  mqttClient.subscribe(Messdaten, subscribeQoS);

  // OLED starten
  oled.begin();

  // Kontrast maximal 255
  oled.setContrast(200);
  oled.setFont(u8g2_font_helvR24_tf);

  // Zeichenfarbe weiß
  oled.setDrawColor(1);

  // horizontale Schrift
  oled.setFontDirection(0);
}

void loop()
{
  // Broker abfragen
  mqttClient.poll();
}

void empfangeneNachricht(int NachrichtGroesse)
{
  Serial.println("Nachricht empfangen: ");

  // Thema der Nachricht empfangen und anzeigen
  String Nachricht = mqttClient.messageTopic();
  Serial.println(Nachricht + " ");

  // Nachricht als String lesen
  String EmpfangeneNachricht = mqttClient.readString();

  // String Anzeige am / trennen
  int Suche = EmpfangeneNachricht.indexOf("/");
  String Temperatur = EmpfangeneNachricht.substring(0, Suche);
  String Luftfeuchtigkeit = EmpfangeneNachricht.substring(Suche + 1, EmpfangeneNachricht.length());

  // Anzeige im Seriellen Monitor
  Serial.println(Temperatur + "°C");
  Serial.println(Luftfeuchtigkeit + "%");
  Serial.println("---------------------");
  Serial.println();

  // Anzeige OLED
  oled.clearDisplay();
  oled.firstPage();

  do
  {
    oled.setCursor(2, 30);
    oled.print(Temperatur);
```

```
    // Grad-Zeichen
    oled.print((char)176);
    oled.print("C");
    oled.setCursor(2, 60);
    oled.print(Luftfeuchtigkeit + "%");

  }
  while (oled.nextPage());
}
```