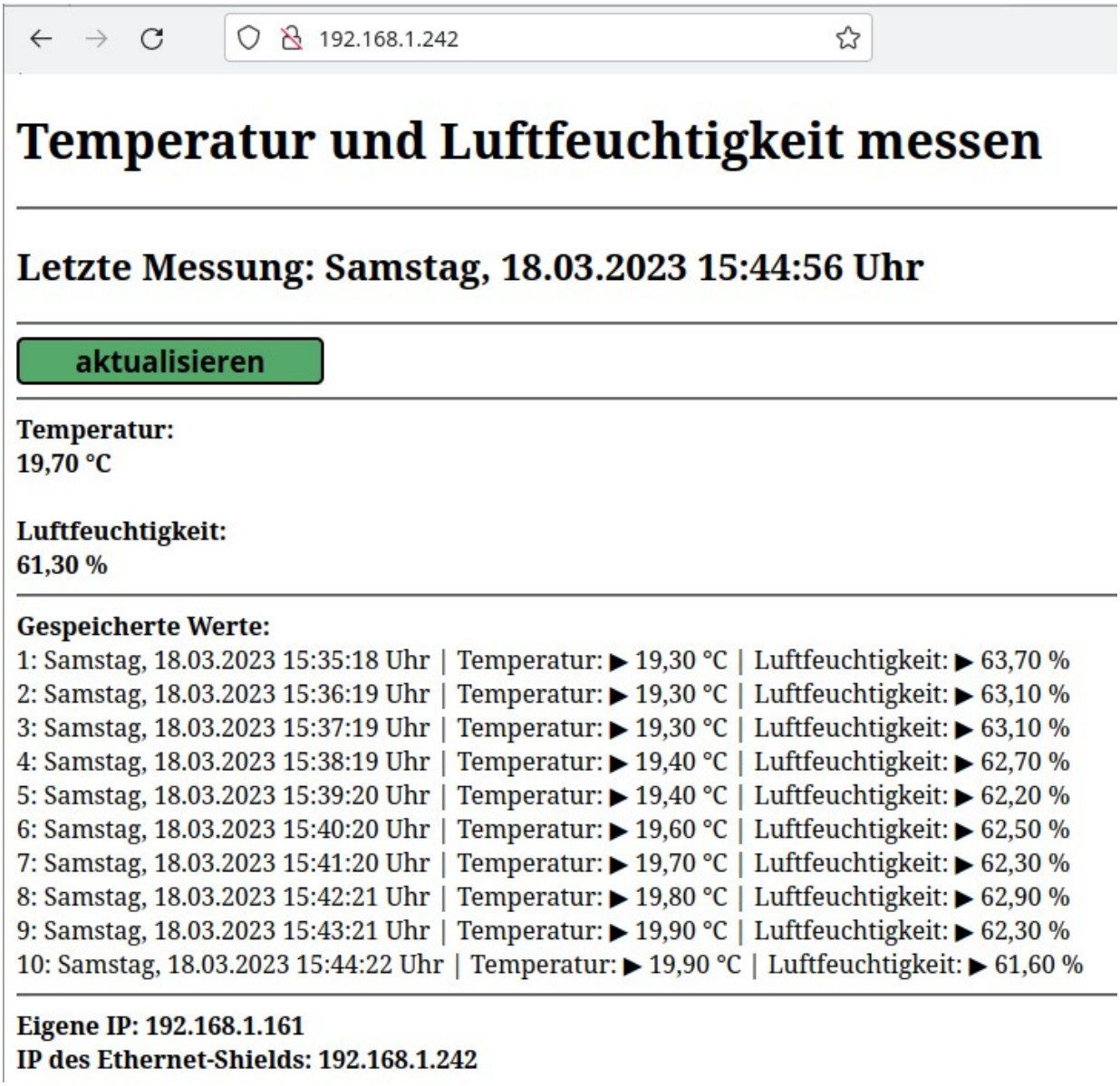
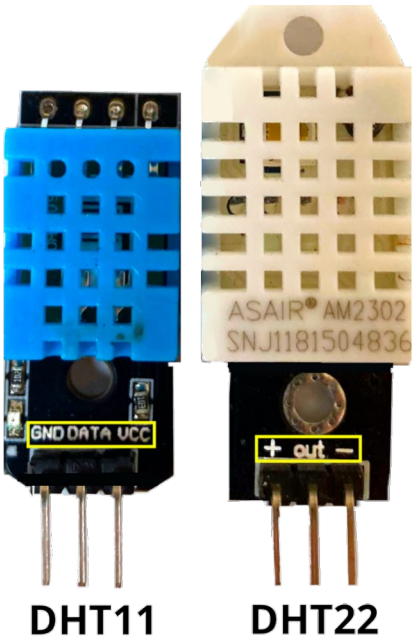
Die mit dem Temperatursensor DHT22 gemessenen Werte sollen in einem Internetbrowser angezeigt werden.

So sieht es aus:

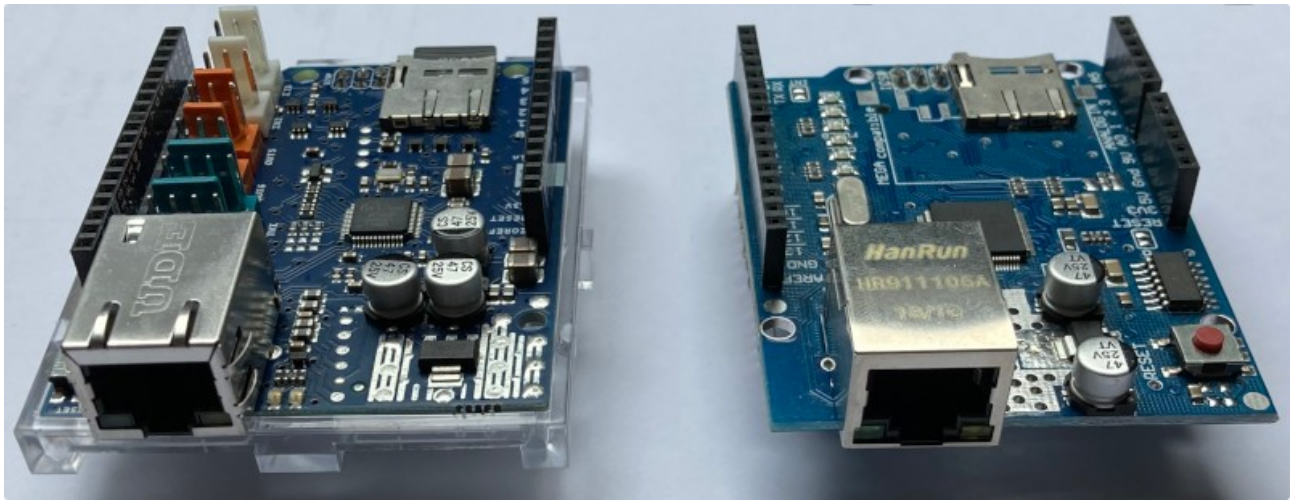


Beispiele für DHT11/DHT22 Sensoren



Die Pinbelegung kann sich von der hier gezeigten unterscheiden. Achte auf die Beschriftung auf dem Modul!

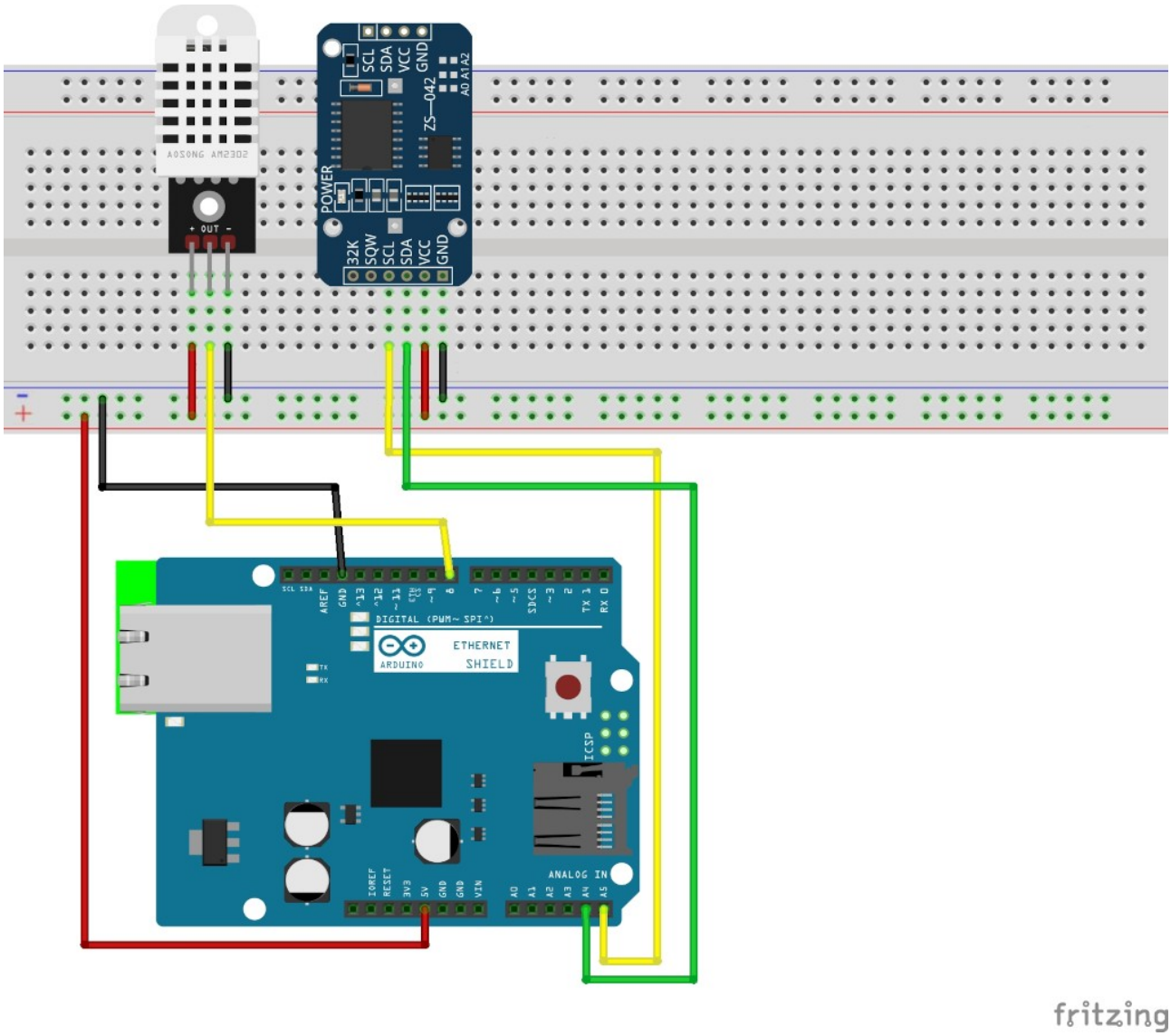
Für diese Anleitung benötigst du ein sogenanntes „Shield“, eine Platine, die einfach auf den Arduino aufgesteckt wird. Auf ihr befindet sich ein LAN-Anschluss (RJ45).



Benötigte Bauteile:

- Temperatursensor DHT11/DHT22
- Ethernet-Shield
- RTC-Modul DS 3231
- Leitungsdrähte

Baue die Schaltung auf:



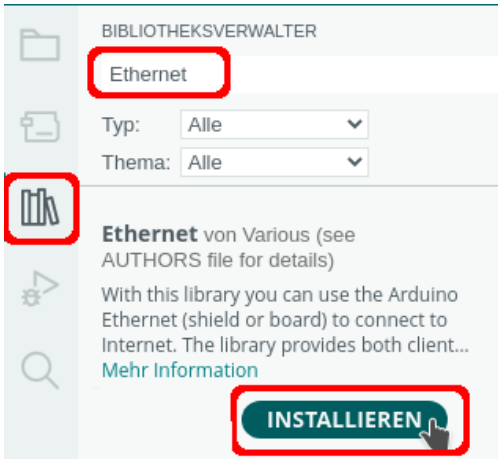
Für das Programm brauchst du eine freie IP-Adresse und eine freie MAC-Adresse in deinem lokalen Netzwerk.
Im Regelfall befindet sich in einem lokalen Netzwerk ein DHCP-Server, der jedem Gerät im Netzwerk automatisch eine IP-Adresse zuteilt. Im Programm wird eine über DHCP vergebene Adresse verwendet.

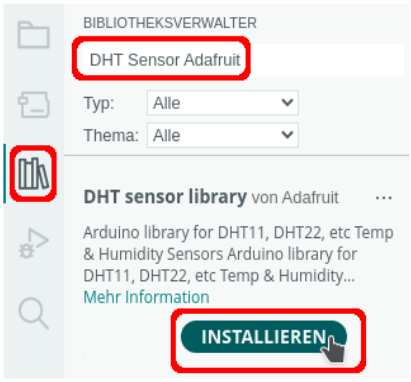
Wenn die Temperaturmessung über einen längeren Zeitraum (mehrere Tage) eingesetzt werden soll, empfiehlt sich die Vergabe einer festen IP-Adresse, weil nach der sogenannten „leasetime“ die IP wieder neu vergeben wird.

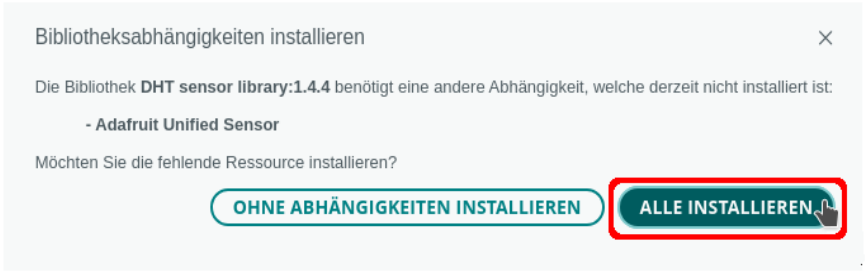
Die MAC-Adresse ist die Hardware-Adresse jeder einzelnen Netzwerkschnittstelle (LAN oder WLAN), mit der jedes Gerät im Netzwerk eindeutig identifiziert werden kann.
Sie besteht aus sechs Bytes in hexadezimaler Schreibweise, die durch „:“ oder „-“ getrennt werden.

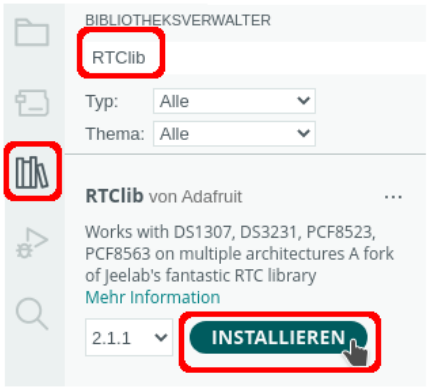
Du kannst die im Programm verwendete „erfundene“ MAC-Adresse übernehmen: Die Gefahr, dass sich ein Gerät mit der gleichen MAC-Adresse im Netzwerk befindet, ist äußerst gering.

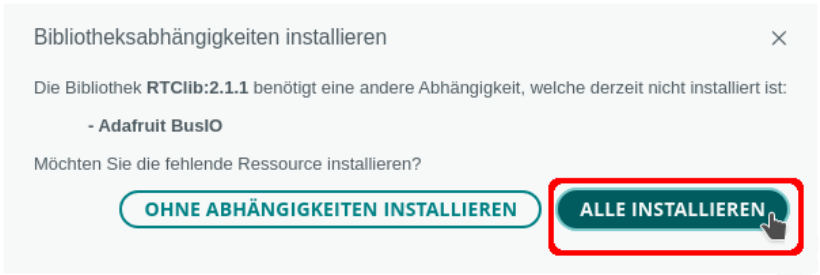
Benötigte Bibliotheken:











Binde die benötigten Bibliotheken ein:

```
# include <Ethernet.h>
# include <DHT.h>
# include <RTClib.h>

// Name des RTC-Moduls
RTC_DS3231 rtc;

// Pin des Sensors
int SENSOR_DHT = 8;

# define SensorTyp DHT11

// oder DHT22
// # define SensorTyp DHT22
```

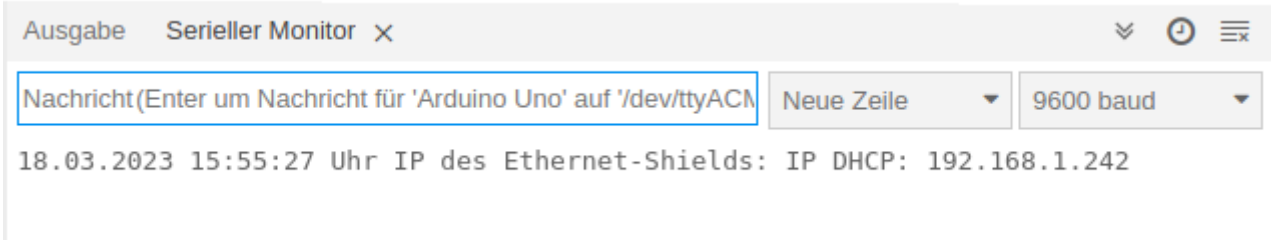
```
// Sensor einen Namen zuweisen
DHT dht(SENSOR_DHT, SensorTyp);

// MAC-Adresse und IP definieren
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

// festeIP = false -> IP-Adresse über DHCP vergeben
bool festeIP = false;

// feste IP
IPAddress ip(192, 168, 1, 200);

// Name des Servers vergeben
EthernetServer Server(80);
byte Zaehler = 0;

// maximale Anzahl der gespeicherten Daten in den Arrays
const byte AnzahlDaten = 11;

// Arrays für die Speicherung
String gespeicherteTemperatur[AnzahlDaten];
String gespeichertesDatum[AnzahlDaten];
String gespeicherteLuftfeuchtigkeit[AnzahlDaten];

// nur jeden zweiten Wert speichern
bool letzterWert = true;
```

Der setup-Teil. Beachte die Kommentare.

```
void setup()
{
  // Temperatursensor starten
  dht.begin();

  // RTC-Modul starten
  rtc.begin();
  /*
    wenn Datum und Zeit nicht korrekt -> Datum/Zeit setzen
    Jahr, Monat, Tag, Stunde, Minute, Sekunde
    keine führende 0 setzen
    Beispiel:
    rtc.adjust(DateTime(2018, 10, 25, 7, 2, 30));
  */
  // rtc.adjust(DateTime(2018, 12, 29, 21, 58, 30));
  Serial.begin(9600);

  // Ethernet starten feste IP
  if (festeIP) Ethernet.begin(mac, ip);

  // Ethernet starten DHCP
  else Ethernet.begin(mac);

  // Server starten
  Server.begin();
  Serial.begin(9600);
  DateTime aktuell = rtc.now();
```

```
  // Datum und Zeit holen
  char Datum[] = "DD.MM.YYYY ";
  char Zeit[] = "hh:mm:ss Uhr";
  Serial.print(aktuell.toString(Datum));
  Serial.print(aktuell.toString(Zeit));

  // zur Kontrolle IP-Adresse anzeigen
  // localIP -> Adresse, die im Browser eingegeben wird
  Serial.print(F(" IP des Ethernet-Shields: "));
  if (festeIP) Serial.print(F("statische IP: "));
  else Serial.print(F("IP DHCP: "));
  Serial.println(Ethernet.localIP());
}
```

Im Seriellen Monitor wird die IP des Ethernet-Shields angezeigt. Diese Adresse wird in einem Browser eingegeben.



Der loop-Teil. Beachte die Kommentare.

```
void loop()
{
  DateTime aktuell = rtc.now();
  float Temperatur;
  float Luftfeuchtigkeit;
  String Nummer;

  // auf Clienten warten ...
  EthernetClient Client = Server.available();

  // neue Anfrage
  if (Client)
  {
    // solange der Client verbunden ist ...
    while (Client.connected())
    {
      if (Client.available())
      {
        char Zeichen = Client.read();
```

```
        // \n = Seite vom Clienten vollständig geladen
        if (Zeichen == '\n')
        {
          // HTTP-Anforderung senden
          Client.println(F("HTTP/1.1 200 OK"));
          Client.println(F("Content-Type: text/html"));
          Client.println("Connection: close");

          // Leerzeile zwingend erforderlich
          Client.println();

          /*
            HTML-Seite aufbauen
            die folgenden Anweisungen müssen
            mit print oder println gesendet werden
            println "verschönert" den Quelltext
            (jede Anweisung in einer eigenen Zeile)
          */
          Client.println(F("<!doctype html>"));
          Client.println(F("<html>"));
          Client.println(F("<body>"));

          // alle 60 Sekunden aktualisieren mit meta-Tag
          Client.println(F("<meta http-equiv=\"refresh\" content=\"60\">"));
          Client.println(F("<h1>Temperatur und Luftfeuchtigkeit messen"));
          Client.println(F("</h1>"));
          Client.println(F("<hr />"));
          Client.print(F("<h2>Letzte Messung: "));

          /*
              Wochentag anzeigen
              0 = Sonntag
              1 = Montag
              ...
              6 = Samstag
          */
          switch (aktuell.dayOfTheWeek())
          {
            case 0:
              Client.print(F("Sonntag"));
              break;
            case 1:
              Client.print(F("Montag"));
              break;
            case 2:
              Client.print(F("Dienstag"));
              break;
            case 3:
              Client.print(F("Mittwoch"));
              break;
            case 4:
              Client.print(F("Donnerstag"));
              break;
            case 5:
              Client.print(F("Freitag"));
              break;
```

```
      case 6:
        Client.print(F("Samstag"));
        break;
    }
    Client.print(", ");

    // Datum und Zeit schreiben
    char Datum[] = "DD.MM.YYYY ";
    char Zeit[] = "hh:mm:ss Uhr";
    Client.print(aktuell.toString(Datum));
    Client.print(F(" "));
    Client.print(aktuell.toString(Zeit));
    Client.println(F("</h2>"));
    Client.println(F("<hr />"));

    // Button anzeigen und formatieren
    Client.println(F("<form>"));
    Client.print(F("<td><input style='font-size:14pt; font-weight:bold;"));
    Client.print(F(" background-color:#55A96B;"));
    Client.print(F(" width:200px; cursor:pointer;"));
    Client.print(F(" border-radius:5px;border: 2px solid black;' type='button'"));
    Client.println(F(" onClick='history.go(0)'"));
    Client.println(F(" value=\"aktualisieren\">"));
    Client.println(F("</form>"));
    Client.println(F("<hr />"));

    // Temperatur lesen
    Temperatur = dht.readTemperature();

    // in String umwandeln, mit replace . durch , ersetzen
    String AnzeigeTemperatur = String(Temperatur);
    AnzeigeTemperatur.replace(".", ",");

    // Luftfeuchtigkeit lesen
    Luftfeuchtigkeit = dht.readHumidity();

    // in String umwandeln, mit replace . durch , ersetzen
    String AnzeigeLuftfeuchtigkeit = String(Luftfeuchtigkeit);
    AnzeigeLuftfeuchtigkeit.replace(".", ",");
    Client.print(F("<b>Temperatur: <br>"));
    Client.println(AnzeigeTemperatur + " &deg;C</b>");
    Client.println(F("<br>"));
    Client.print(F("<br><b>Luftfeuchtigkeit: <br>"));
    Client.println(AnzeigeLuftfeuchtigkeit + " %</b><hr />");

    // Datum speichern
    gespeichertesDatum[Zaehler] = aktuell.toString(Datum);
    gespeichertesDatum[Zaehler] = gespeichertesDatum[Zaehler] + aktuell.toString(Zeit);

    // Temperatur speichern
    gespeicherteTemperatur[Zaehler] = AnzeigeTemperatur;

    // Luftfeuchtigkeit speichern
    gespeicherteLuftfeuchtigkeit[Zaehler] = AnzeigeLuftfeuchtigkeit;
    Client.println(F("<b>Gespeicherte Werte:<br></b>"));
```

```
// Daten anzeigen
for (int i = 0; i <= Zaehler - 1; i ++)
{
  Client.println(String(i + 1) + ": ");
  switch (aktuell.dayOfTheWeek())
  {
    case 0:
      Client.print(F("Sonntag"));
      break;
    case 1:
      Client.print(F("Montag"));
      break;
    case 2:
      Client.print(F("Dienstag"));
      break;
    case 3:
      Client.print(F("Mittwoch"));
      break;
    case 4:
      Client.print(F("Donnerstag"));
      break;
    case 5:
      Client.print(F("Freitag"));
      break;
    case 6:
      Client.print(F("Samstag"));
      break;
  }
  Client.print(", ");
  Client.println(gespeichertesDatum[i]);
  Client.print(F(" | Temperatur:  &#9654; "));
  Client.println(gespeicherteTemperatur[i] );
  Client.print(F(" &deg;C" ));
  Client.print(F(" | Luftfeuchtigkeit:  &#9654; "));
  Client.println(gespeicherteLuftfeuchtigkeit[i]);
  Client.println(F("%"));
  Client.println(F("<br>"));
}
Client.println(F("<hr />"));

// IPs anzeigen
Client.print(F("<b>Eigene IP: "));
Client.print(Client.remoteIP());
Client.println(F("<br>"));
Client.println(F("IP des Ethernet-Shields: "));
Client.print(Ethernet.localIP());
Client.println(F("</b>"));
Client.println(F("</body>"));
Client.println(F("</html>"));

// Zeit, um die Antwort zu übertragen
delay(1);

// Verbindung beenden
Client.stop();
```

```
          // Zaehler erhöhen und bei > AnzahlDaten zurücksetzen
          if (Zaehler < AnzahlDaten && letzterWert) Zaehler ++;
          if (Zaehler >= AnzahlDaten) Zaehler = 0;
          letzterWert = !letzterWert;
        }
      }
    }
  }
}
```