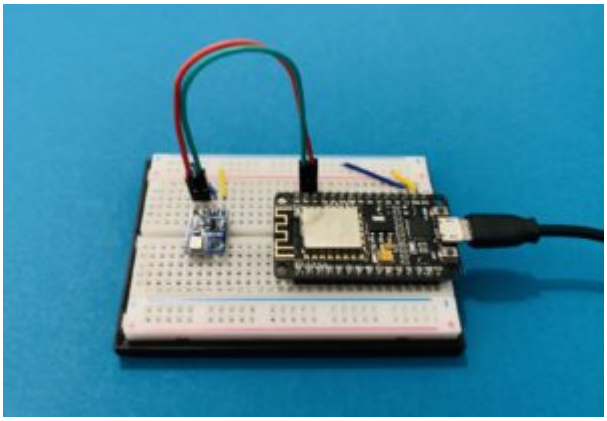
Überwache die Temperatur mit Telegram und einem ESP8266



In diesem Projekt überwachst du die Temperatur mit einem Sensor und deinem ESP8266. Sobald der Temperatursensor einen von dir festgelegten Wert ermittelt, sendet dein Microcontroller eine Nachricht an deinen Telegram-Bot.

Dieses Projekt ist der zweite Teil einer Serie: Inhaltlich baut es auf unserem Stillen Alarm mit Telegram auf. Schaue dort hinein, um mehr über die grundlegenden Funktionen des Sketchs zu erfahren. Auch in diesem Projekt wartet dein ESP8266 darauf, dass ein bestimmtes Ereignis eintritt. Allerdings wird er hier nicht durch einen Interrupt getriggert, sondern fragt selbst die Daten eines Temperatursensors ab.

Anfänger

1 – 2 Stunden

ca. 12 €

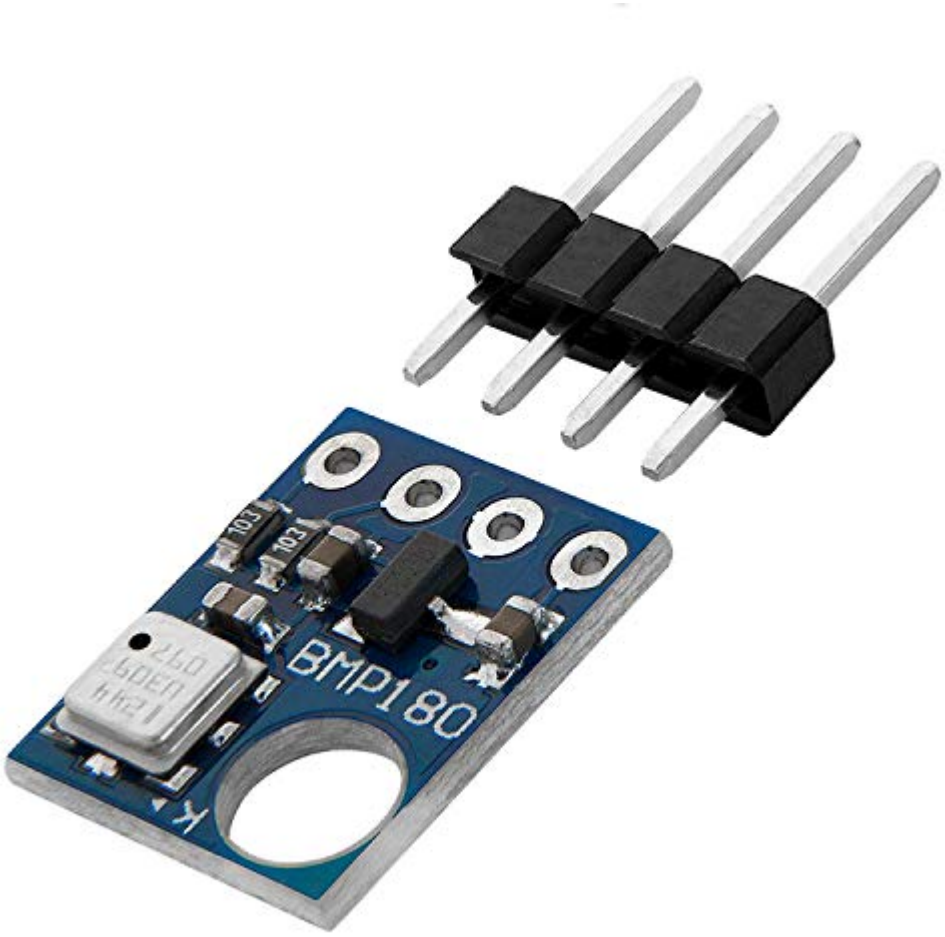
Für dieses Projekt benötigst du (Mengen s. Beschreibung):

[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 €

Angebot

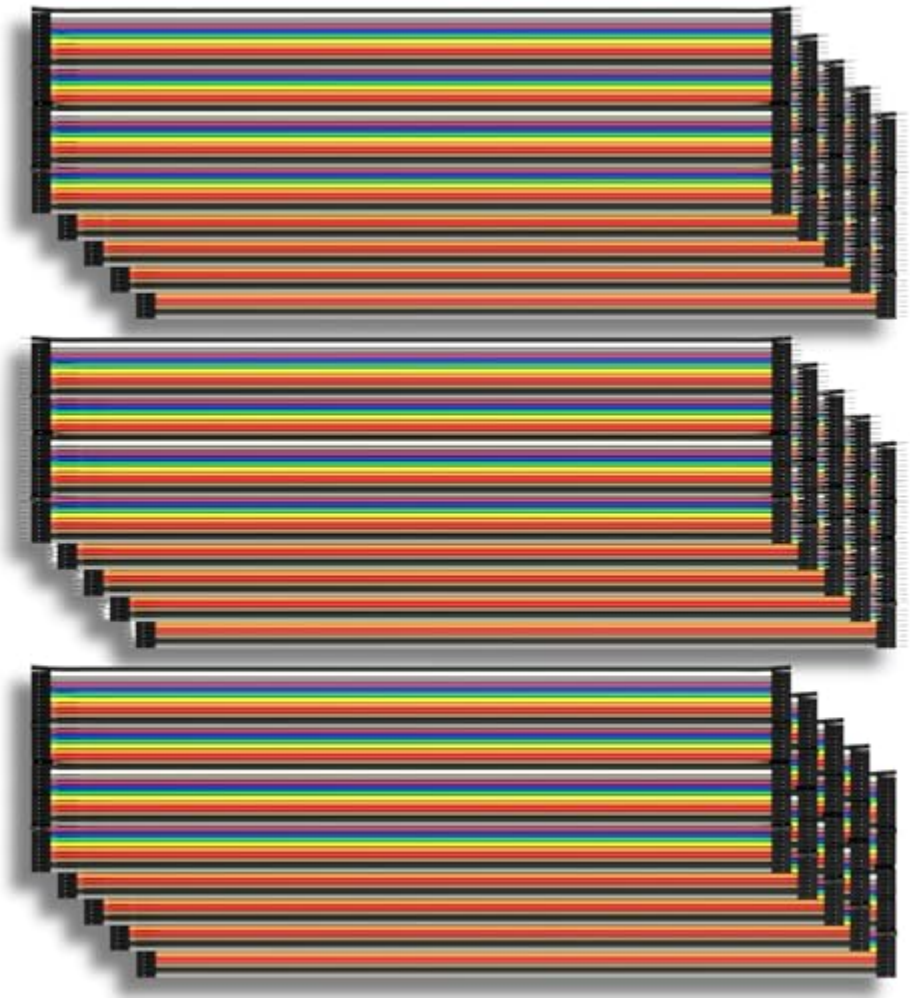


[AZDelivery GY-68 BMP180 Barometrischer Luftdruck und Temperatur Sensor kompatibel mit Arduino und Raspberry Pi inklusive E-Book!]
4,49 €

[AZDelivery Mini Breadboard 400 Pin mit 4 Stromschienen kompatibel mit Arduino und Jumper Wire Kabeln]

 Breadboard; Steckbrett für schnellen Aufbau elektronischer Schaltungen mit 400...

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 €

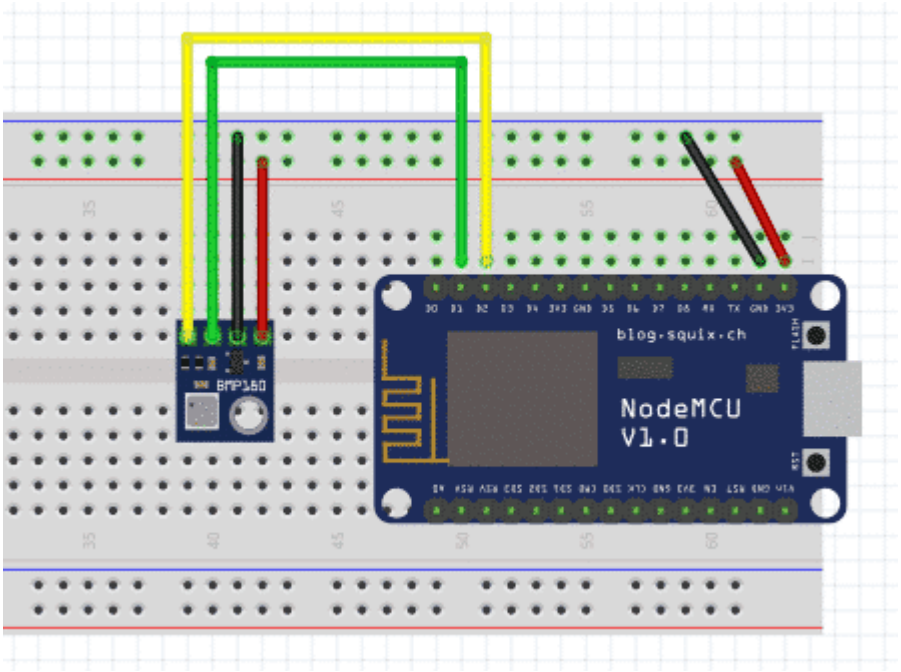
Der passende Temperatursensor

In diesem Projekt verwenden wir den Sensor BMP180. Du kannst aber natürlich auch jeden anderen Temperatursensor verwenden – z.B. einen einfachen TMP36, einen DHT22 oder einen GY-906. Denke in diesem Fall daran, deinen Sketch entsprechend anzupassen.

In diesen Tutorials lernst du, wie du einen TMP36 und einen GY-906 anschließt und verwendest.

Der Aufbau des Projekts

Du benötigst nur deinen ESP8266, den Temperatursensor (in unserem Fall einen BMP180), ein Breadboard und Kabel. Orientiere dich beim Aufbau an diesem Schema:



Der Sensor BMP180 wird per I²C angeschlossen. Am ESP8266 musst du deshalb zwingend die beiden Pins **D1** und **D2** verwenden. Schließe den Sensor wie folgt an:

BMP180	ESP8266
VIN	3v3
GND	GND
SDA	D2
SCL	D1

Der Sketch

___STEADY_PAYWALL___

Kopiere den folgenden Sketch in deine Arduino IDE, ergänze

deine Daten und lade ihn auf deinen ESP8266.

Sketch als .txt anschauen

```
/*
   Die Temperatur überwachen mit Telegram - polluxlabs.net
*/

//Bibliotheken
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>

#include <Wire.h>
#include <Adafruit_BMP085.h>

// Deine WLAN-Zugangsdaten
const char* ssid = "NETZWERKNAME";
const char* password = "PASSWORT";

// Den Telegram-Bot initialisieren
#define botToken "DEIN TOKEN"  // den Bot-Token bekommst du vom Botfather)

//Deine UserID
#define userID "DEINE USERID"

WiFiClientSecure client;
UniversalTelegramBot bot(botToken, client);

Adafruit_BMP085 bmp;

//Variablen für die Temperatur
float temp;
float threshold = 27.00;

//Verbindung zum WLAN
void connectToWiFi() {
  Serial.print("Verbinde mich mit: ");
  Serial.println(ssid);
```

```
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(300);
  }
  Serial.println("");
  Serial.println("Verbunden!");
}

void setup() {
  Serial.begin(115200);
  client.setInsecure();

  if (!bmp.begin()) {
      Serial.println("Kein Sensor gefunden! Checke die
Verbindung.");
    while (1) {}
  }
}

void loop() {
  temp = bmp.readTemperature();
  Serial.print("Temperatur = ");
  Serial.print(temp);
  Serial.println(" °C");

  if (temp > threshold) { //Die Temperatur, die überschritten
werden muss
    connectToWiFi();
      bot.sendMessage(userID, "Temperatur zu hoch!   " +
String(temp) + " °C", "");
    Serial.println("Temperatur zu hoch!");
    WiFi.disconnect();
    delay(300000);
  }
  delay(500);
}
```

Was ist neu in diesem Sketch?

Vieles im Code haben wir schon im Stillen Alarm verwendet. Ein paar Teile sind jedoch neu und diese schauen wir uns nun genauer an.

Zunächst benötigst du zwei weitere Bibliotheken. **Wire.h** ist für die Kommunikation per I²C zuständig. Diese Bibliothek ist standardmäßig bereits vorinstalliert. Damit du den Sensor BMP180 so einfach wie möglich verwenden kannst, existiert ebenfalls eine passende Bibliothek: **Arduino_BMP085.h**. Lass dich nicht vom Namen irritieren. Dort steht zwar **BMP085** (das ist der Vorläufer des BMP180), sie funktioniert jedoch auch mit dem BMP180 problemlos.

Am Anfang deines Sketchs bindest du also zusätzlich diese beiden Bibliotheken ein:

```
#include <Wire.h>
#include <Adafruit_BMP085.h>
```

Ebenfalls zu Beginn des Sketchs erstellst du für den Sensor das Objekt **bmp** sowie zwei Variablen: eine für die gemessene Temperatur (temp) und eine für den Schwellenwert (threshold), bei dessen Überschreiten dein ESP8266 eine Nachricht an dich sendet. In unserem Beispiel setzen wir diesen Schwellenwert auf 27,00 °C. Da der BMP180 die Temperatur als Kommazahl ausgibt, benötigst du für diese Variablen den Dateityp **float**.

```
Adafruit_BMP085 bmp;
float temp;
float threshold = 27.00;
```

In der Setup-Funktion prüfst du, ob der Temperatursensor von deinem ESP8266 gefunden wurde und verwendet werden kann. Ist das nicht der Fall, wirst du im Seriellen Monitor darüber informiert und der Sketch begibt sich mit **while(1)** in eine Endlosschleife – friert also ein.

```
  if (!bmp.begin()) {
      Serial.println("Kein Sensor gefunden! Checke die
Verbindung.");
    while (1) {}
  }
```

Solltest den obigen Text also in deinem Seriellen Monitor zu Gesicht bekommen, trenne deinen ESP8266 vom Strom und überprüfe deine Verkabelung. In den meisten Fällen sollte hier der Fehler liegen.

Der Loop

Im Loop fragst du die Temperatur im Halbsekundentakt ab und prüfst mit einem Schwellenwert, ob sie diesen überschritten hat.

```
temp = bmp.readTemperature();
```

Wenn das der Fall ist, verbindet sich dein ESP8266 mit dem Internet und sendet eine entsprechende Nachricht an dein Smartphone.

```
if (temp > threshold) { //Wenn die Temperatur über dem
Schwellenwert liegt
    connectToWiFi();
     bot.sendMessage(userID, "Temperatur zu hoch!  " +
String(temp) + " °C", "");
```

Die Funktion **bot.sendMessage()** besteht aus drei Teilen. Der mittlere ist die Nachricht selbst, die du hier jedoch aus zwei Strings und der Variablen **temp** „zusammenbaust“. Hierfür verbindest du die einzelnen Teilstrings einfach mit einem Pluszeichen – doch Vorsicht: In der Mitte befindet sich die Variable **temp** als **float**. Um den Wert in dieser Variablen senden zu können, musst du diese erst mit der Funktion **String(temp)** in einen String umwandeln.

Auf deinem Smartphone erscheint dann der Text: „Temperatur zu hoch! x °C“ – wobei das **x** für die gemessene Temperatur steht.

Zuletzt trennt dein ESP8266 die Verbindung zu deinem WLAN wieder und setzt sich für 5 Minuten (300.000 Millisekunden) zur Ruhe. Erst dann beginnt er wieder damit, alle 500 Millisekunden die Temperatur vom BMP180 abzufragen.

```
    WiFi.disconnect();
    delay(300000);
  }
  delay(500);
}
```

Sollte danach die Temperatur immer noch über dem Schwellenwert liegen, erhältst du eine weitere Nachricht.

Falls noch nicht geschehen, lade den Sketch auf deinen ESP8266 und probiere ihn gleich aus.

Das passende Gehäuse

Möchtest du das Projekt in einem kleinen Gehäuse unterbringen? Hier findest du passende 3D-Druck-Dateien, die du herunterladen und ausdrucken kannst.

Wie geht es weiter?

Im nächsten Teil der Serie wartest du nicht darauf, bis dein ESP8266 sich bei dir meldet, sondern fragst die Temperatur selbst von deinem Smartphone aus ab.

Statt eines Temperatursensors kannst du auch andere Sensoren einsetzen: So kannst du dich warnen lassen, wenn die Luft zu schlecht oder das Licht zu hell ist. Es gibt auch Sensoren, die Flammen erkennen können – solltest du diesen einsetzen, hoffen wir, dass du niemals eine entsprechende Nachricht erhältst!