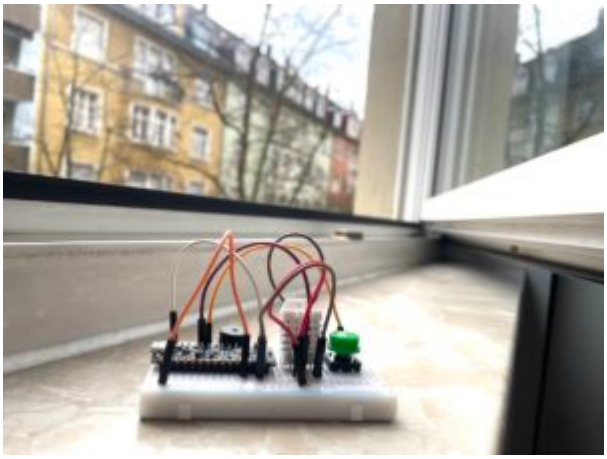
# Die Zimmer lüften mit dem Arduino



Gehörst du zu den Leuten, die gerne die Zimmer lüften, aber dann vergessen, dass das Fenster offen ist? Mir passiert das regelmäßig – bis jetzt. **In diesem Projekt überwacht ein Arduino Nano die Temperatur, sobald du das Fenster zum Lüften öffnest.** Fällt die Temperatur im Raum um 0,5°C schlägt er Alarm. So kannst du sicher sein, dass der Raum oder sogar die ganze Wohnung nicht unnötig auskühlt.

**Diese Bauteile benötigst du:**

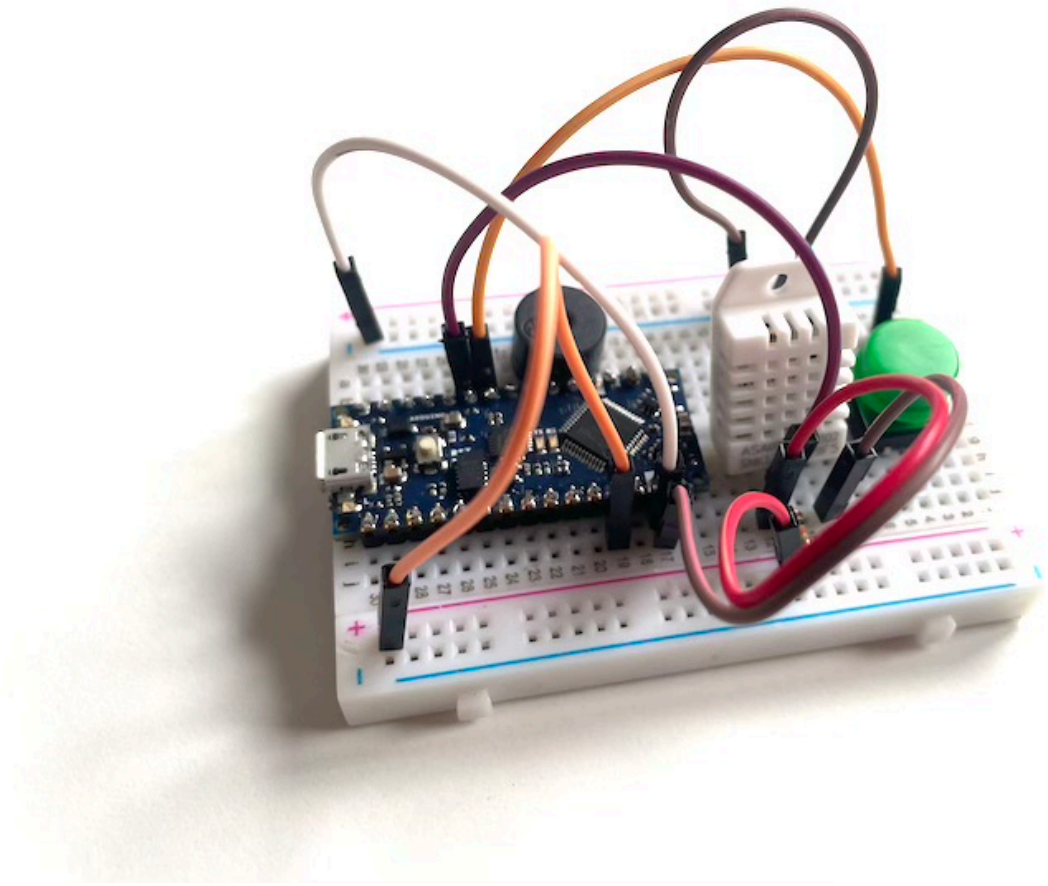
- Arduino Nano
- Temperatursensor DHT22
- Piezo-Summer
- Button
- 2x 10kΩ Widerstände
- Breadboard & Kabel

## So funktioniert die

# Temperaturüberwachung

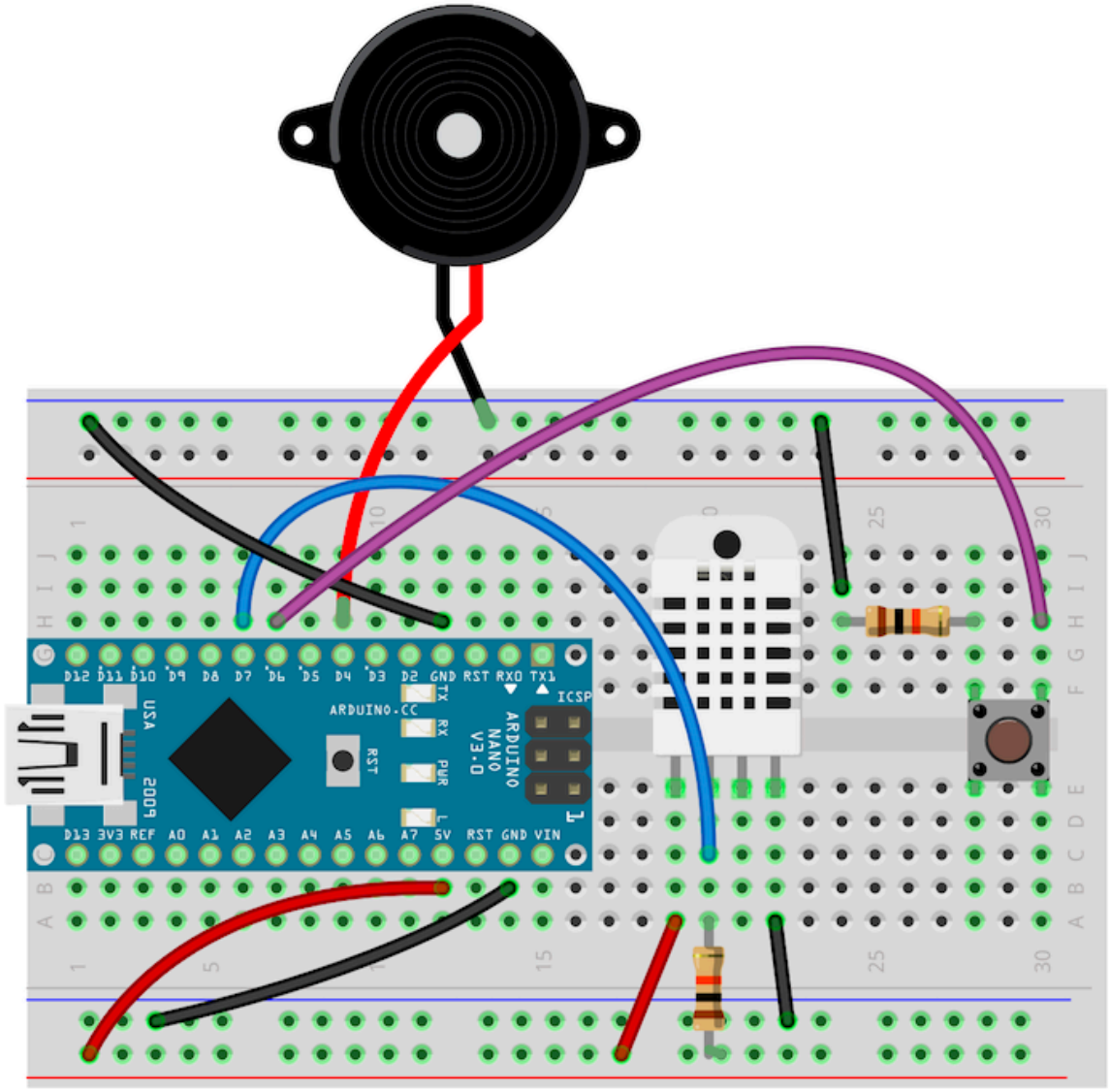
Wenn du das Fenster öffnest, drückst du einen Button am Arduino – damit wird der Startwert der Temperatur festgelegt und die Messung beginnt. Während nun (mal abgesehen vom Sommer) kühlere Luft ins Zimmer gelangt, fällt die Temperatur darin kontinuierlich. Sobald die Zimmertemperatur um 0,5°C gefallen ist, schlägt der Piezo-Summer Alarm und ruft dich ans geöffnete Fenster, um es zu schließen. Du drückst den Button erneut: der Alarm hört auf und der Arduino wird wieder in den Wartezustand zurückgesetzt – bis zum nächsten Lüften.

Natürlich kannst du den Wert von 0,5°C ändern und so einstellen, dass er zu deinen Wünschen, zur Positionierung des Arduinos im Zimmer etc. passt.



# So baust du das Projekt zusammen

Orientiere dich beim Aufbau an der folgenden Skizze. Ich habe für dieses Projekt wegen seiner kompakten Größe einen Arduino Nano verwendet. Du kannst aber natürlich auch ein anderes Board verwenden, das du gerade zur Hand hast – also zum Beispiel einen Arduino UNO. Auch muss es nicht unbedingt der Sensor DHT22 sein – in diesem Tutorial lernst du, wie du den „kleinen Bruder“ DHT11 am Arduino anschließt und verwendest.




Wenn du alles verkabelt hast, kann es mit dem Sketch weitergehen.

# Der Sketch für das Projekt

Bevor du den folgenden Sketch kopierst und auf deinen Arduino hochlädst, benötigst du noch die passenden Bibliotheken für den Sensor DHT22 – falls du in der Arduino IDE noch nie ein Projekt mit diesem Sensor umgesetzt hast.

Öffne in diesem Fall den Bibliotheksmanager in der IDE und suche nach **DHT sensor library** und klicke auf Installieren. In einem Dialogfenster wirst du daraufhin gefragt, ob du die Bibliothek **Adafruit Unified Sensor** gleich mitinstallieren möchtest. Bestätige das mit einem Klick auf Ja.

Nun zum vollständigen Sketch:



```
#include <DHT.h>

// Pins
#define DHTPIN 7        // DHT22 an Pin D7
#define BUZZER_PIN 4    // Piezo-Summer an Pin D4
#define BUTTON_PIN 6    // Taster an Pin D6
#define LED_PIN 13      // Interne LED an Pin D13

// DHT Sensor
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

// Variablen
float startTemperature = 0.0;
bool measuring = false;
bool alarmActive = false;

void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(BUZZER_PIN, OUTPUT);
   pinMode(BUTTON_PIN, INPUT_PULLUP); // Interner Pull-Up-
Widerstand aktivieren
  pinMode(LED_PIN, OUTPUT);
```

```
  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(LED_PIN, LOW); // LED initial ausgeschaltet
}

void loop() {
  // Taster prüfen (wechseln zwischen Messung und Leerlauf)
  if (digitalRead(BUTTON_PIN) == LOW) { // Taster gedrückt
    delay(50); // Entprellung
    // Sicherstellen, dass der Taster weiterhin gedrückt ist
    if (digitalRead(BUTTON_PIN) == LOW) {
      if (!measuring) {
        // Messung starten
        measuring = true;
         alarmActive = false; // sicherstellen, dass der Alarm
deaktiviert ist
        startTemperature = dht.readTemperature();
        if (isnan(startTemperature)) {
          Serial.println("Fehler beim Lesen des DHT22!");
          startTemperature = 0.0; // Standardwert
        }
        Serial.print("Messung gestartet. Starttemperatur: ");
        Serial.println(startTemperature);
        digitalWrite(LED_PIN, HIGH); // LED einschalten
      } else {
        // In Leerlauf wechseln: Alarm und Messung stoppen
        noTone(BUZZER_PIN); // Alarmton stoppen
        digitalWrite(LED_PIN, LOW); // LED ausschalten
        alarmActive = false;
        measuring = false;
          Serial.println("Alarm ausgeschaltet und Leerlauf
aktiviert.");
      }
      // Warten bis der Button losgelassen wird
      while (digitalRead(BUTTON_PIN) == LOW);
      delay(50); // Entprellzeit nach Loslassen
    }
  }
  // Messlogik, wenn aktiv
  if (measuring) {
    float currentTemperature = dht.readTemperature();
    if (isnan(currentTemperature)) {
```

```
      Serial.println("Fehler beim Lesen des DHT22!");
      delay(2000);
      return;
    }

    // Temperaturänderung prüfen
    if (currentTemperature <= startTemperature - 0.5) {
      alarmActive = true; // Alarm auslösen
    }

    // Alarm ausführen
    if (alarmActive) {
      tone(BUZZER_PIN, 1000); // Alarmton
      Serial.println("Alarm: Temperatur ist gesunken!");
    } else {
      noTone(BUZZER_PIN); // Alarmton deaktivieren
    }

    // Debug-Ausgabe
    Serial.print("Aktuelle Temperatur: ");
    Serial.println(currentTemperature);

    delay(2000); // Messung alle 2 Sekunden
  }
}
```

## So funktioniert der sketch

Lass uns einen genaueren Blick auf den Sketch werfen. Zunächst bindest du die Bibliothek für den Temperatursensor ein und legst die Anschlüsse der Bauteile fest:

```
#include <DHT.h>

#define DHTPIN 7       // DHT22 an Pin D7
#define BUZZER_PIN 4   // Piezo-Summer an Pin D4
#define BUTTON_PIN 6   // Taster an Pin D6
#define LED_PIN 13     // Interne LED an Pin D13
```

```
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

Weiter geht es mit den benötigten Variablen. In **startTemperature** wird die Temperatur zu Beginn deiner Messung – also sobald du auf den Button drückst – erfasst. Die beiden Variablen **measuring** und **alarmActive** dienen später der Programmsteuerung.

```
float startTemperature = 0.0;
bool measuring = false;
bool alarmActive = false;
```

## Setup-Funktion

Hier startest du den Seriellen Monitor und initialisierst den Sensor. Außerdem legst du die benötigten **pinModes** fest und aktivierst für den Button den internen Pullup-Widerstand des Arduinos. Zuletzt stellst du sicher, dass der Piezo-Summer und die interne LED ausgeschaltet sind.

```
void setup() {

  Serial.begin(9600);
  dht.begin();
  pinMode(BUZZER_PIN, OUTPUT);
   pinMode(BUTTON_PIN, INPUT_PULLUP); // Interner Pull-Up-Widerstand aktivieren
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(LED_PIN, LOW); // LED initial ausgeschaltet
}
```

## Loop-Funktion

Das ist der Hauptteil des Programms, der kontinuierlich läuft. Hier wird entschieden, ob die Messung durchgeführt wird oder nicht:

```
if (digitalRead(BUTTON_PIN) == LOW) {

  // Wurde der Button gedrückt? Dann beginnt die Messung, falls sie nicht bereits läuft.

}
```

Bei einem Druck auf den Button passiert Folgendes: Wenn die Messung nicht läuft (die Variable **measuring** ist **false**), wird sie gestartet. Die Variable **measuring** wird dann auf **true** gesetzt.

Wenn die Messung allerdings läuft, wird sie gestoppt. Der Alarm wird deaktiviert und die LED ausgeschaltet. Die Variable **measuring** wird wieder auf **false** gesetzt. Der Arduino befindet sich nun sozusagen im Leerlauf.

In der Messung selbst wird alle 2 Sekunden geprüft, ob die aktuelle Temperatur mehr als 0,5°C unterhalb der Starttemperatur liegt. Ist das der Fall, wird der Alarm aktiviert:

```
// Temperaturänderung prüfen
if (currentTemperature <= startTemperature - 0.5) {
  alarmActive = true; // Alarm auslösen
}
```

Hierfür wird die Variable **alarmActive** auf **true** gesetzt, was dazu führt, dass in der darauffolgenden Abfrage der Piezo-Summer aktiviert wird:

```
if (alarmActive) {
  tone(BUZZER_PIN, 1000); // Alarmton
  Serial.println("Alarm: Temperatur ist gesunken!");
} else {
  noTone(BUZZER_PIN); // Alarmton deaktivieren
}
```

Hier wird geprüft, ob **alarmActive == true** ist (in einer verkürzten Schreibweise) und der Piezo entsprechend aktiviert oder deaktiviert.

## Wie geht es weiter?

Du hast nun einen Arduino, der laut Alarm schlägt, sobald die Temperatur entsprechend deiner Einstellung gefallen ist. Statt einen Piezo-Summers kannst du aber natürlich auch eine andere Benachrichtigung wählen. Wie wäre es z.B. mit einer Push Notification auf deinem Smartphone? Hierfür benötigst du allerdings einen Microcontroller mit WLAN, wie z.B. den ESP8266 oder ESP32.