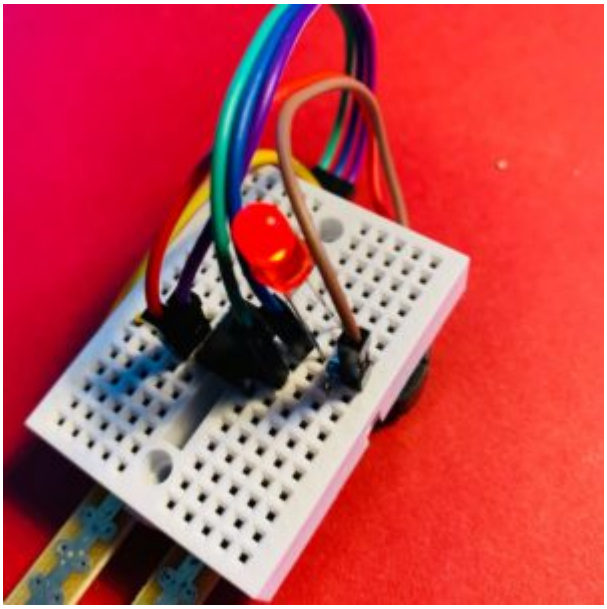
# Kompakter Pflanzenwächter mit Batterie

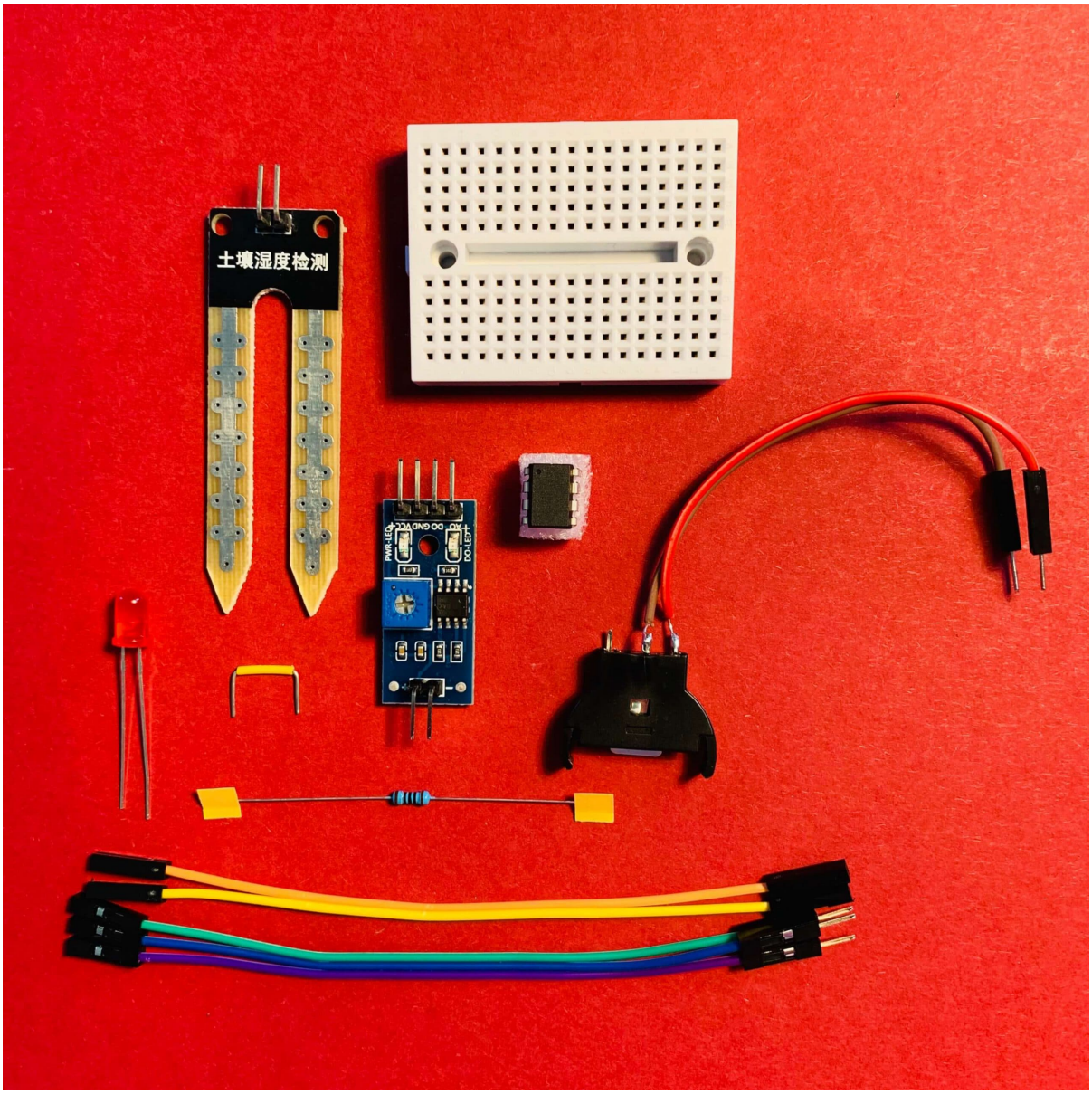


Hast du einen grünen Daumen? Selbst wenn, dieses Projekt erinnert dich daran, deine Pflanzen zu gießen. Stecke den Pflanzenwächter einfach neben deine Lieblingspflanze in ihren Topf und kalibriere ihn auf die richtige Feuchtigkeit. Sobald die Erde zu trocken geworden ist, fängt die LED an zu blinken und erinnert dich rechtzeitig an die Gießkanne.

Für dieses Projekt benötigst du:

- ATtiny85*
- Feuchtigkeitssensor*
- 3,3V Knopfzelle (CR 2032)
- Batteriehalter* mit verlöteten Kabeln
- LED*
- 100Ω Widerstand*
- Kabel
- Mini-Breadboard*

**Amazon Affiliate Link – wenn du dort bestellst, erhalten wir eine kleine Provision.*



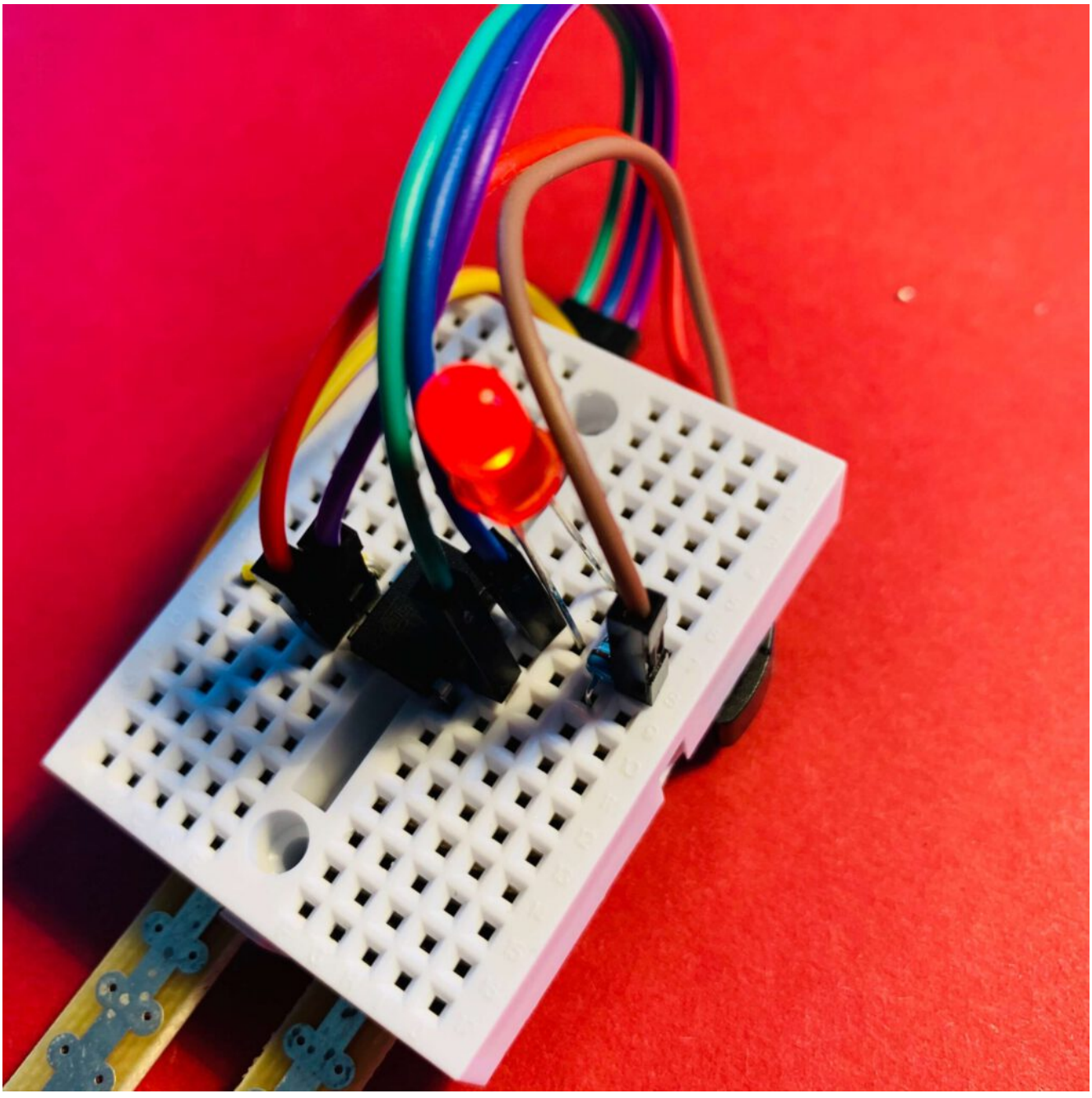

# Das kann der Pflanzenwächter

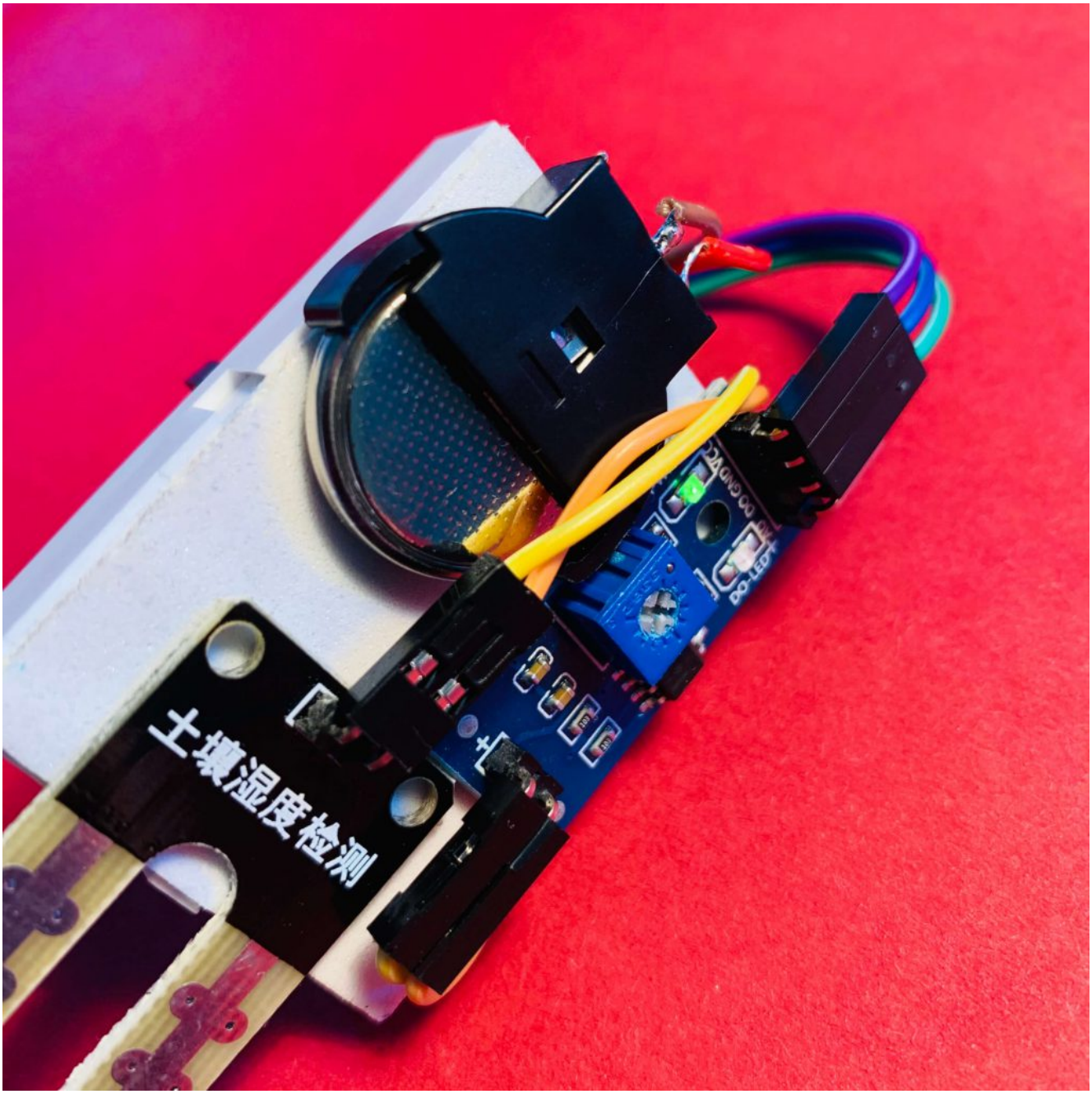
Ein guter Pflanzenwächter sollte in einem Topf neben der Pflanze nicht allzu sehr auffallen und außerdem eine unabhängige Stromversorgung haben. Hierfür bietet sich der ATtiny85 an, denn **diesen kannst du mit einer 3,3V Knopfzelle über eine sehr lange Zeit betreiben** – vorausgesetzt du lässt ihn nicht ununterbrochen messen, sondern versetzt ihn die meiste Zeit in den Schlafmodus.

Mit dem Code dieses Projekts testest du nur alle 30 Minuten,

ob die Pflanzenerde zu trocken geworden ist. Die restliche Zeit schläft der Pflanzenwächter und zieht nur sehr wenig Strom von der Batterie. So sollte diese ohne Probleme mehrere Monate durchhalten.

**Damit der Wächter eine kompakte Große erhält, verwendest du Vorder- und Rückseite des Mini-Breadboards.** Auf die Rückseite klebst du einige Bauteile, die du mit Kabeln auf der Vorderseite mit dem ATtiny85 verbindest. Der fertige Pflanzenwächter misst in seiner Box zum Schluss nur ca. 40 x 70 x 40 mm.






## So programmierst du den ATtiny85

Bevor du mit dem Zusammenbau loslegst, musst du den passenden Sketch auf deinen ATtiny85 laden. Den Code findest du am Ende dieses Projekts. Erfahre hier, wie du ihn mit Hilfe eines Arduino Uno auf den ATtiny85 hochlädst.

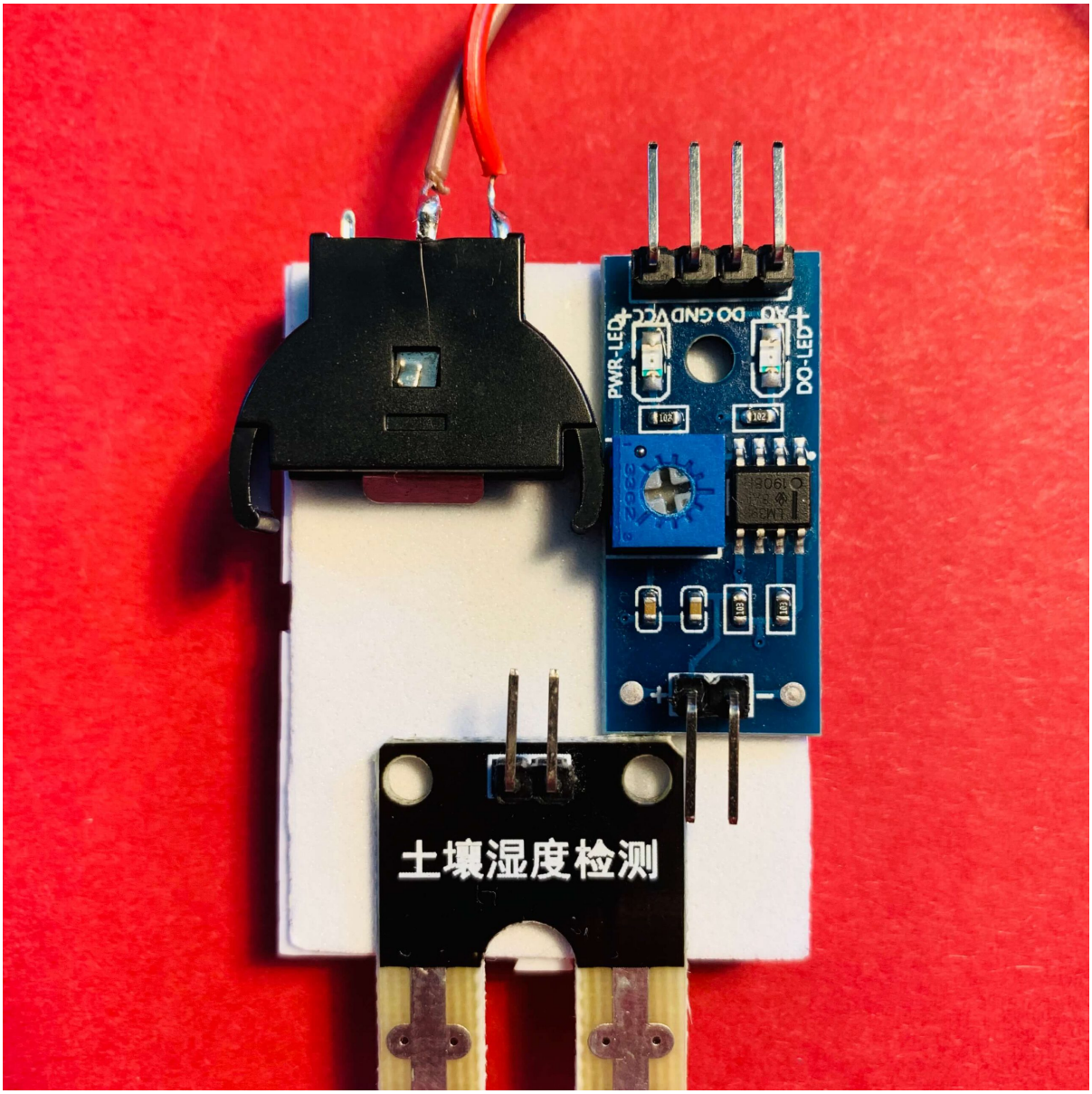
# So baust du den Pflanzenwächter zusammen

Der Aufbau dieses Projekts dauert ungefähr 30 Minuten und ist auch von Anfängern zu bewerkstelligen.

## Zunächst die Rückseite

Praktischerweise befindet sich auf der Rückseite des Breadboards eine Klebefläche, auf die du einige Bauteile anbringen kannst. Ziehe also zunächst die Schutzfolie von der Rückseite ab.

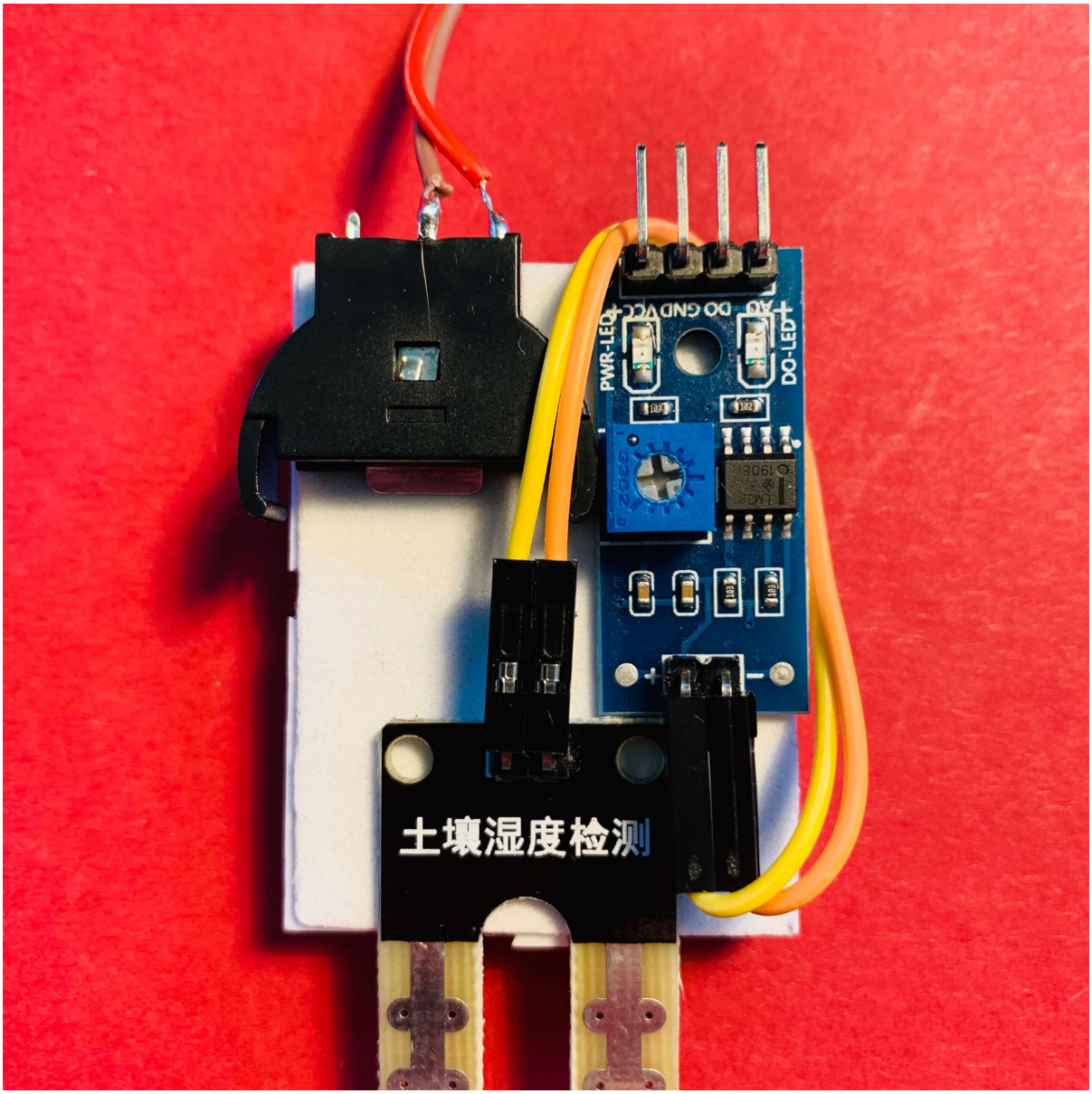
Klebe anschließend den Batteriehalter sowie Board und Sonde des Feuchtigkeitssensors wie folgt auf die Fläche. Drücke alle Teile zunächst nur leicht an. Wenn alles passt, kannst du sie stärker andrücken – sie halten dann sehr gut auf der Rückseite.



Auf die Rückseite geklebte Bauteile

## Die Kabel an der Rückseite anbringen

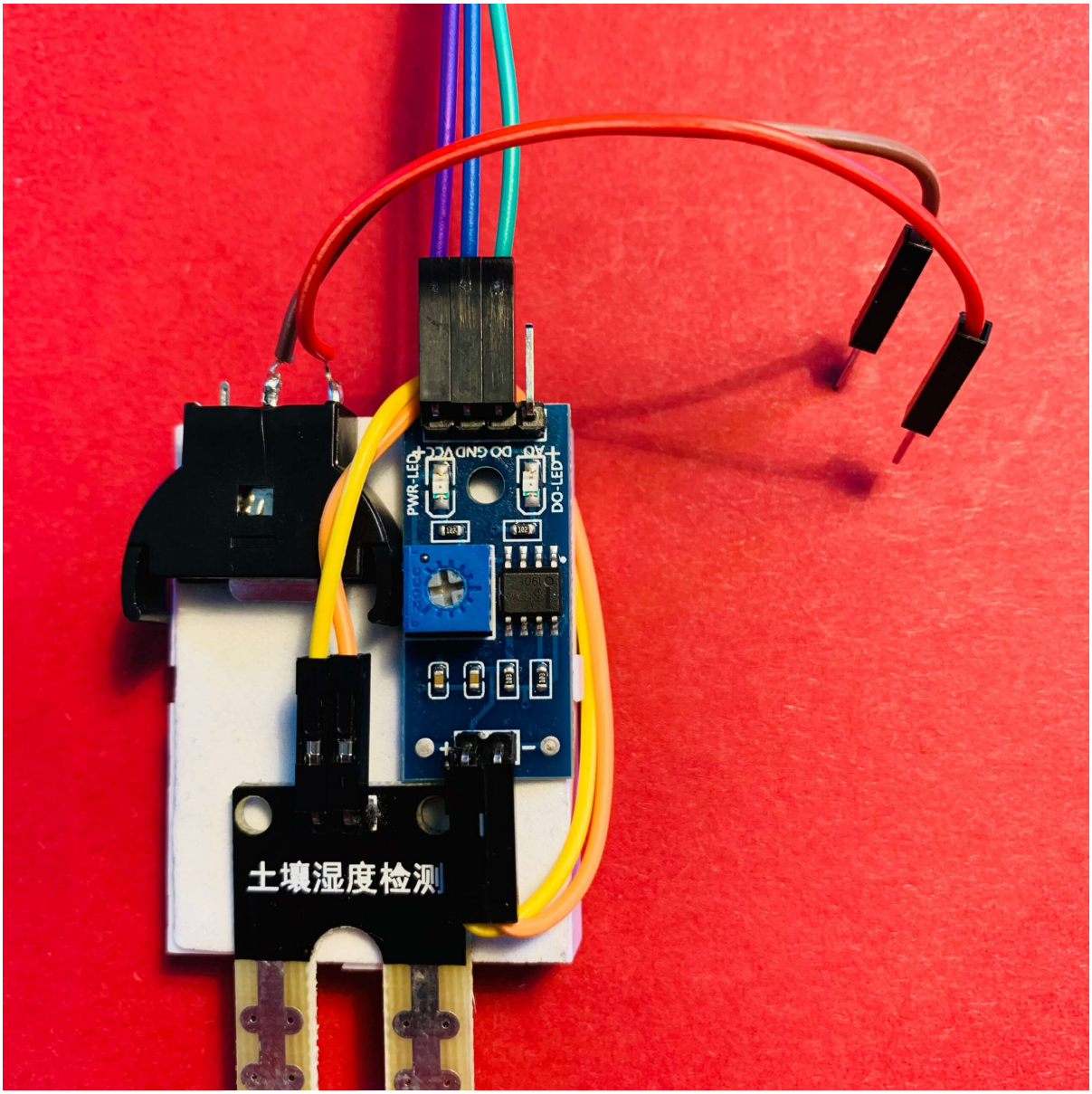
Bleiben wir zunächst auf der Rückseite. Hier musst du zunächst die Sonde mit dem Board verbinden. Hierfür benötigst du die beiden Kabel, die an beiden Enden Buchsen haben. Bringe sie wie folgt an:




Wenn du die Buchsen an die Pins gesteckt hast, lege das Kabel um die Oberseite des Breadboards. So nehmen sie am wenigsten Platz weg.

Jetzt kommt das Board des Feuchtigkeitssensors an die Reihe. An ihm findest du noch vier freie Pins: VCC (Plus), GND (Minus), DO (Digitalausgang) und AO (Analogausgang). Du benötigst allerdings nur die ersten drei.

Stecke also zunächst drei Kabel an die Pins VCC, GND und DO.




Damit hast du die Rückseite des Pflanzenwächters vorerst abgeschlossen – Zeit für die Vorderseite des Breadboards!

## Den ATtiny85 auf das Board setzen

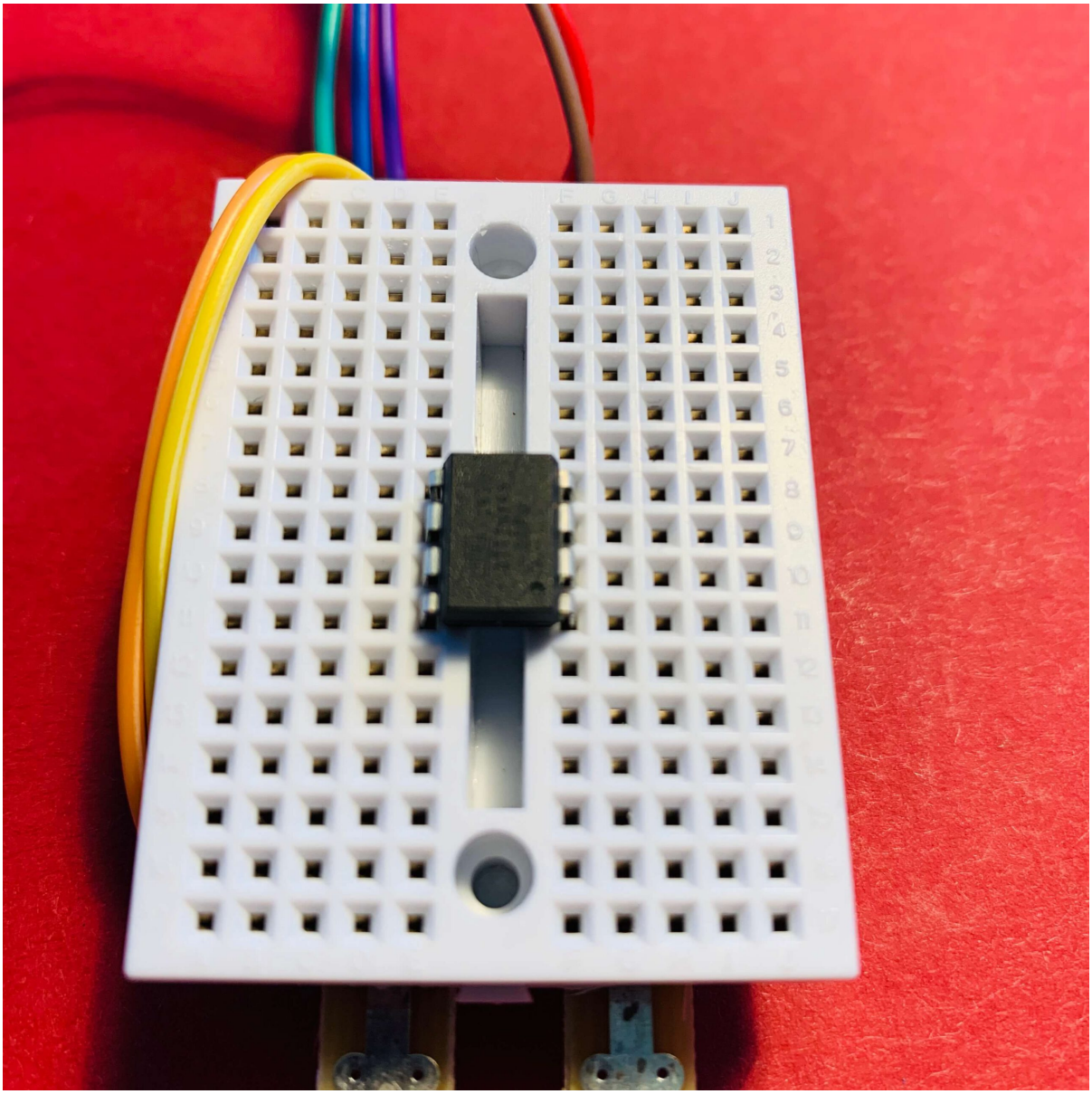
Nimm deinen ATtiny85 in die Hand und schau ihn dir genau an: In einer Ecke des Chips findest du eine kleine Runde Vertiefung. Diese markiert den Pin neben ihr als Nummer Eins.

Lege das Breadboard nun so vor dich hin, dass die Sonde des Feuchtigkeitssensor nach unten zeigt. Setze den ATtiny85 jetzt **vorsichtig** in die Mitte des Breadboard, so dass die kleine Vertiefung nach rechts unten zeigt. Vorsichtig deshalb, weil

die Pins des Chips sehr sensibel sind. **Sei also besonders vorsichtig, wenn du ihn aufsetzt oder wieder vom Breadboard nimmst.**

> *Verwende am besten einen Sockel für deinen ATtiny85, um ihn zu beim Ein- und Ausbau zu schonen.*

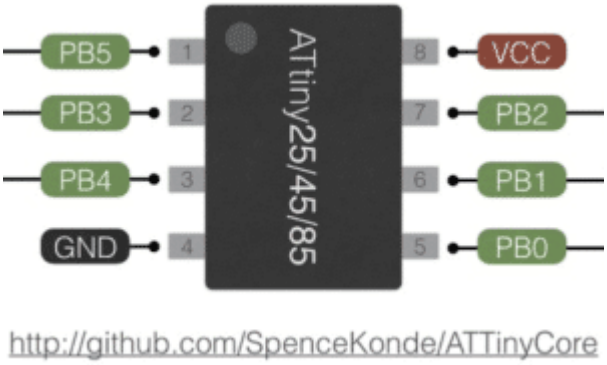
So sollte das aussehen, wenn du fertig bist:



## Verbinde den Feuchtigkeitssensor mit dem

## ATtiny85

Bevor du die Kabel des Feuchtigkeitssensors auf das Breadboard steckst, schaue dir zunächst einmal die Nummerierung der Pins des ATtiny85 an:




Pinout des ATtiny85

Wie gesagt, markiert die kleine Vertiefung den Pin 1 des ATtiny85. Die Nummern der anderen Pins erhältst du, wenn du von hier aus gegen den Uhrzeigersinn zählst.

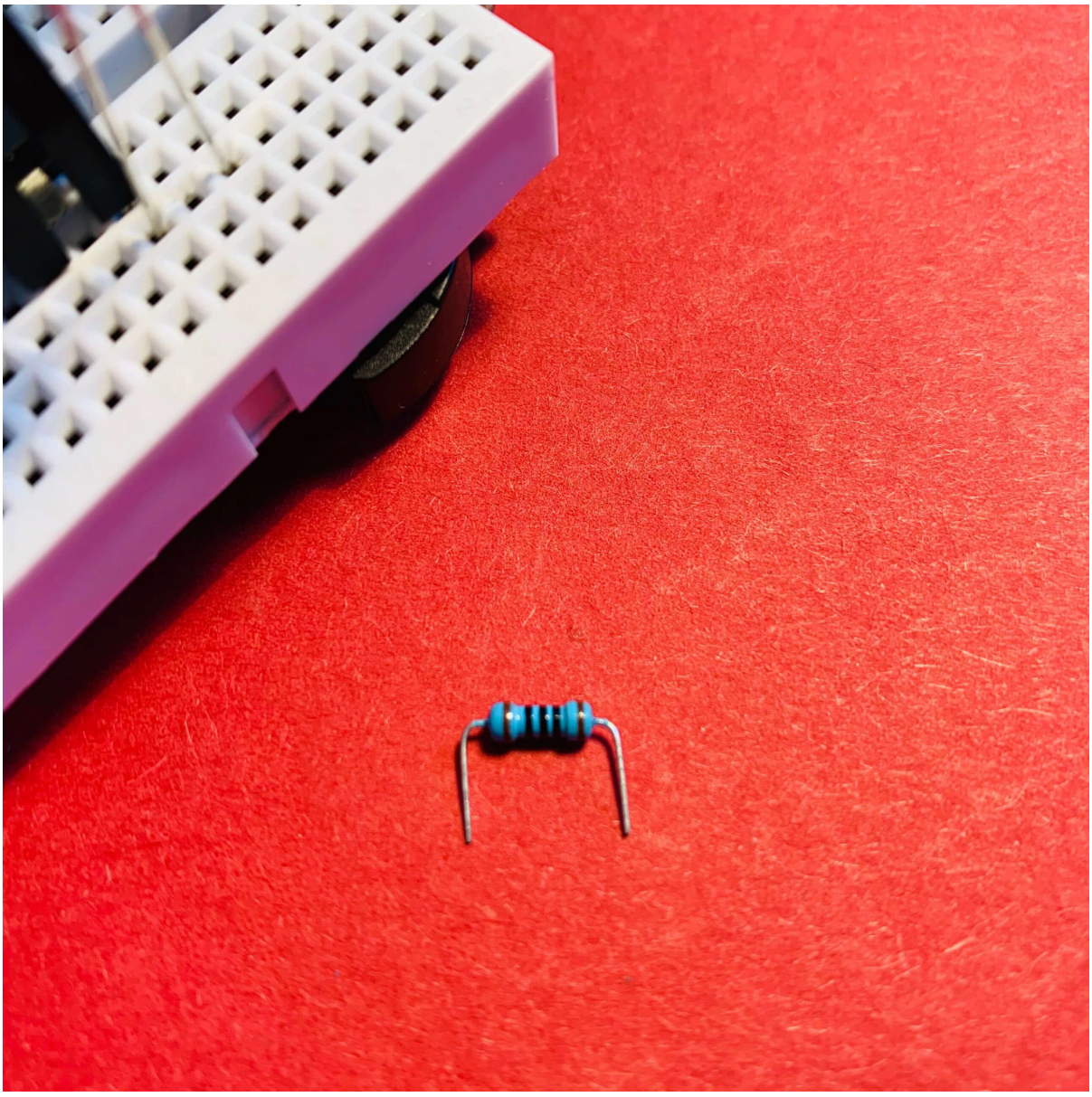
Drehe dein Breadboard nun am besten, sodass die Sonde des Feuchtigkeitssensors nach oben zeigt. So zeigt dein ATtiny85 in die gleiche Richtung wie in der Darstellung oben.

Stecke jetzt das Kabel von Pin DO des Sensors direkt neben Pin 2 des ATtiny85. GND kommt an Pin 4 – das ist der Minuspol. Den Pin VCC des Sensors steckst du zuletzt neben Pin 7 des ATtiny85. Hier noch einmal in der Übersicht:

| Pins Feuchtigkeitssensor | Pins ATtiny85 |
|---|---|
| DO | 2 |
| GND | 4 |
| VCC | 7 |

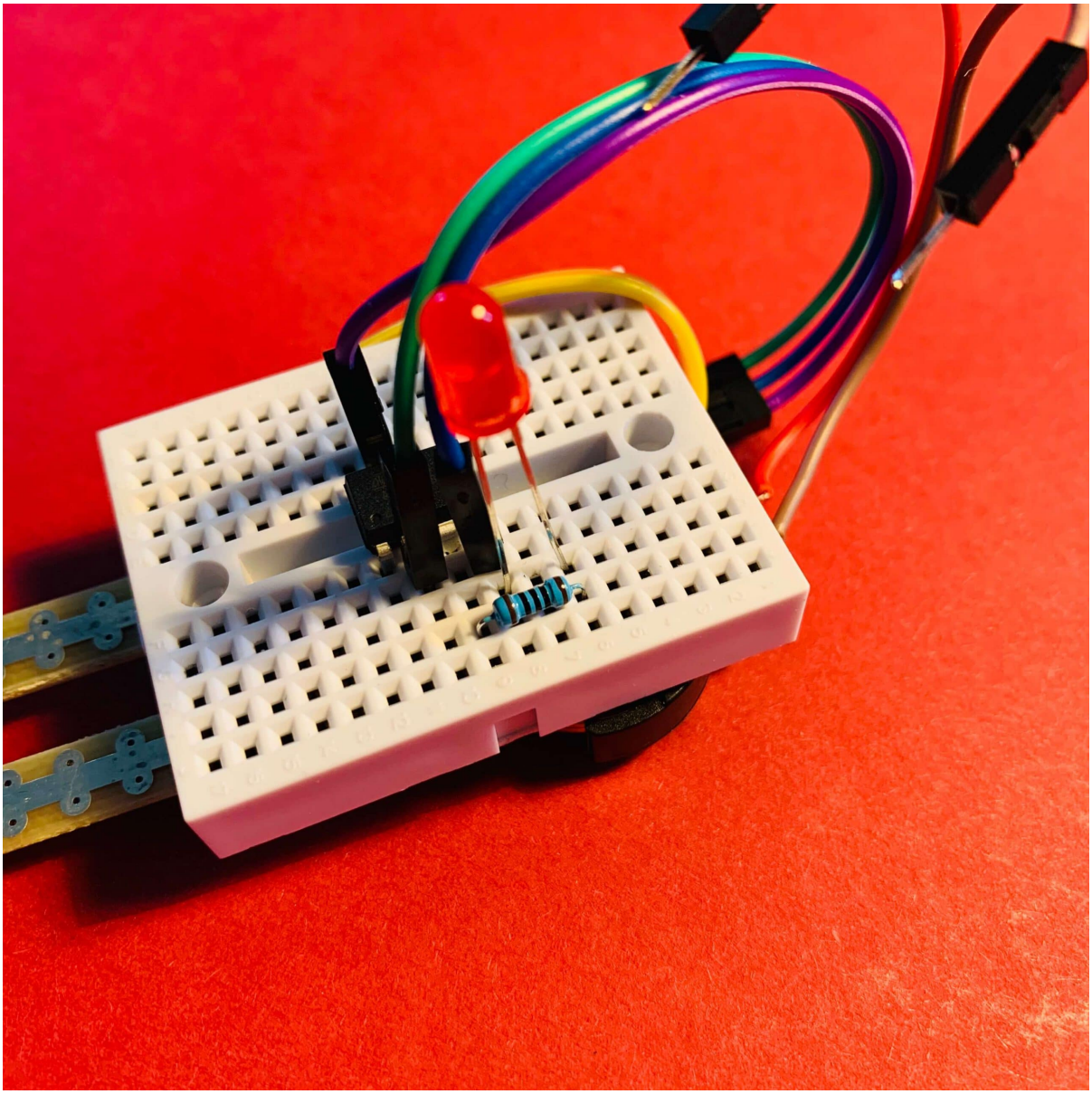
## Bringe die LED samt Widerstand an

Bevor du den Widerstand auf das Breadboard setzt, musst du ihn zurechtbiegen und die Beine kürzen. So sollte der Widerstand nach deiner Behandlung aussehen:



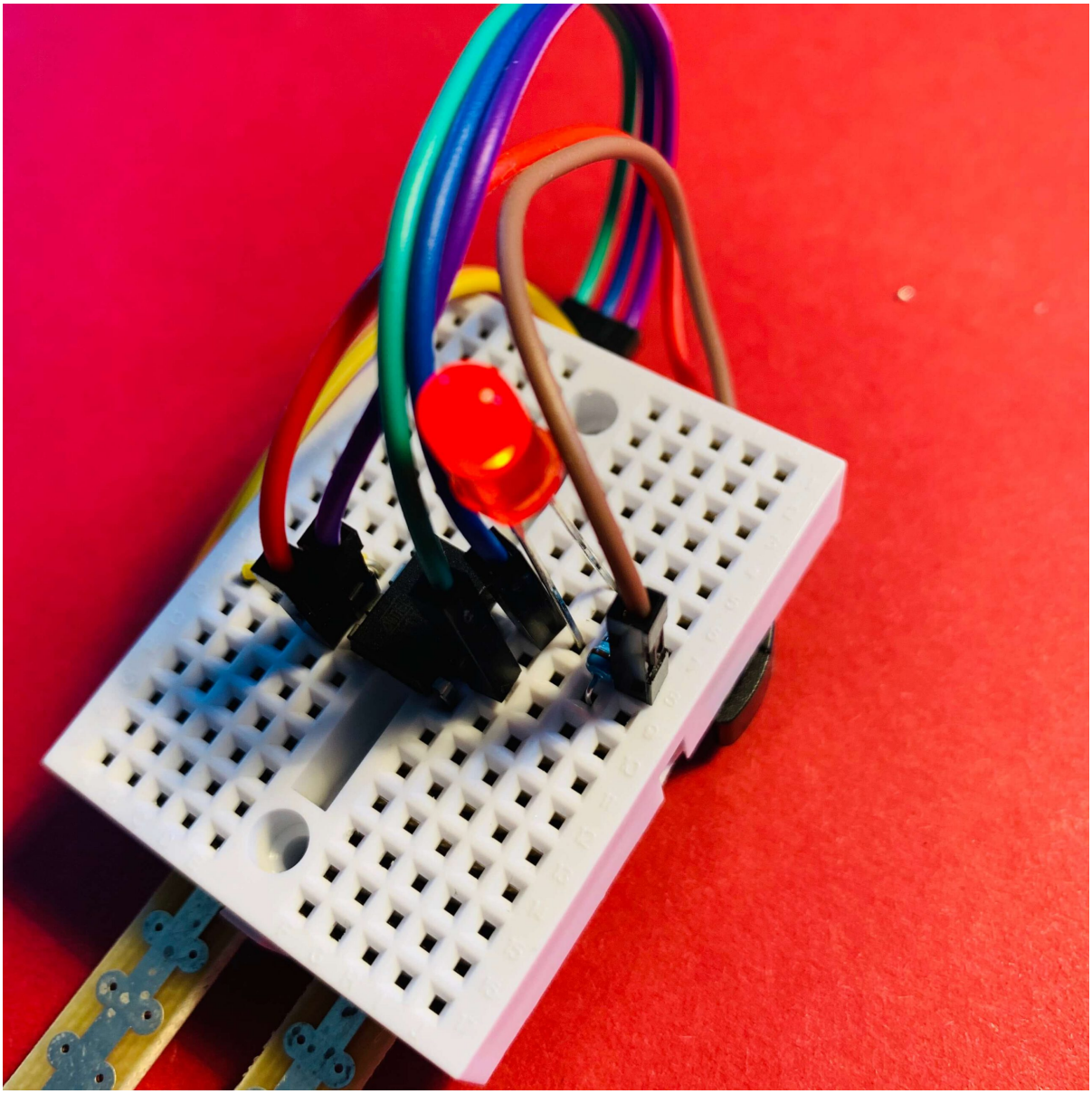
Nimm nun die LED und stecke ihr kurzes Bein (Kathode) neben das Kabel an Pin 4 des ATtiny85. Ihr langes Bein (Anode) kommt zwei Löcher daneben.

Stecke nun den Widerstand wieder eine Reihe weiter, sodass er die Anode mit Pin 3 des ATtiny85 verbindet. So sollte das aussehen:



## Versorge den Pflanzenwächter mit Strom

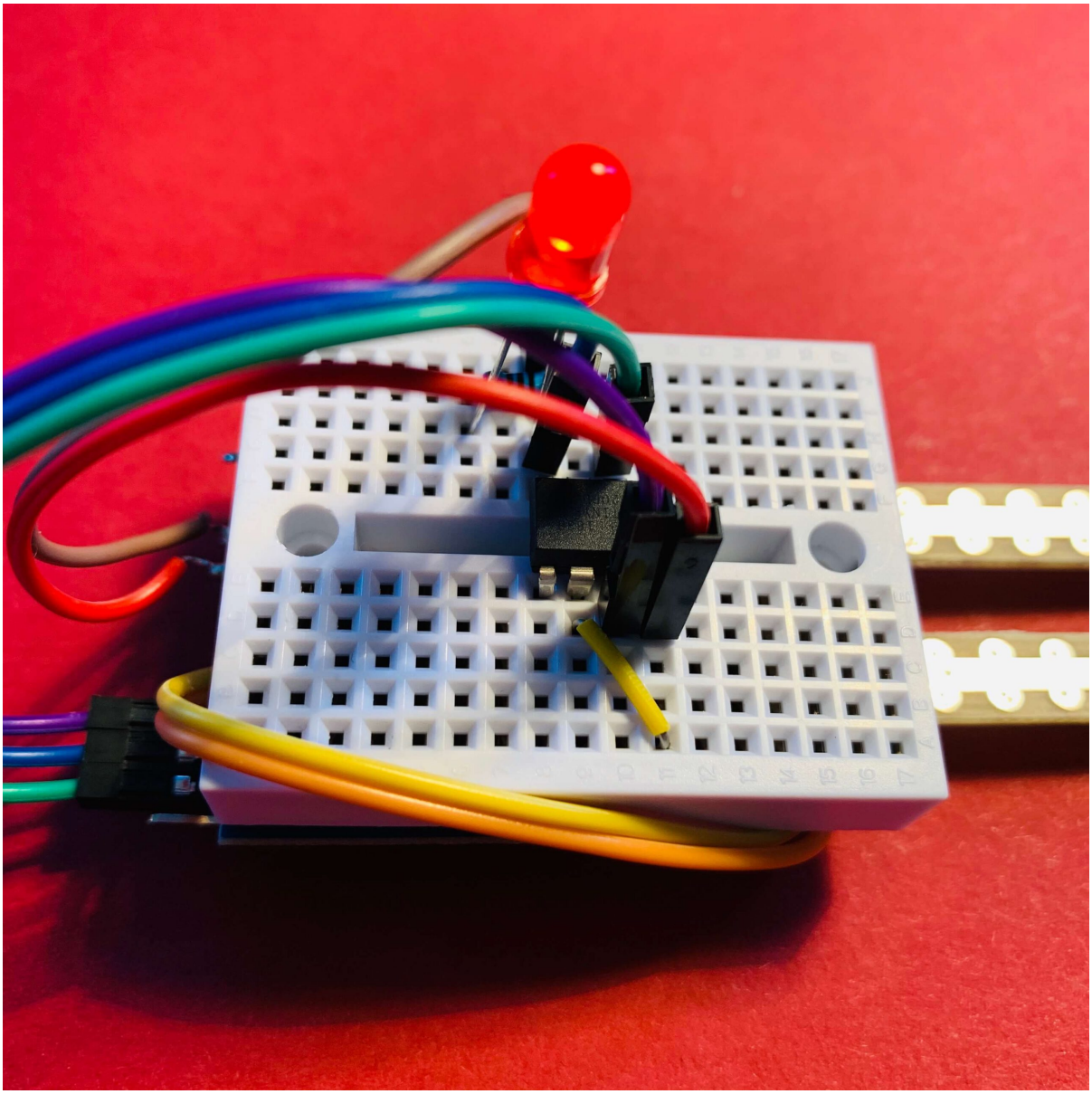
Jetzt der wichtigste Teil: die Stromversorgung. Nimm die beiden Kabel des Batteriehalters in die Hand uns stecke das rote Kabel neben Pin 8 des ATtiny85 und das schwarze in das letzte freie Loch neben Pin 4:



## Der letzte Draht

Jetzt brauchst du noch eine kleine Drahtbrücke. Diese benötigst du, um den Pflanzenwächter zu kalibrieren, sodass er Alarm schlägt, wenn die Erde um ihn herum zu trocken ist.

Stecke diese Drahtbrücke nun zwischen Pin 8 und Pin 6 des ATtiny85:



**Und das war es!** Dein Pflanzenwächter ist zusammengebaut und bereit für die Kalibrierung.

## So kalibrierst du deinen Pflanzenwächter

Bevor du loslegst, drehe mit Hilfe eines Schraubenziehers den Poti an der Rückseite des Feuchtigkeitssensors in beide Richtungen. Wenn du alles richtig miteinander verbunden hast, sollte die LED an- und ausgehen – je nachdem wie weit du den Poti nach links und rechts drehst.

Wenn sie das tut, kann es weitergehen:

Setze deinen Pflanzenwächter mit der Sonde neben eine Pflanze in die Erde. **Am besten ist diese Erde gerade so trocken, dass du die Pflanze jetzt eigentlich gießen würdest.** Drehe den Poti an der Rückseite so dass die LED gerade noch brennt. Und das war es auch schon – die Kalibrierung ist abgeschlossen und der Wächter weiß, wann er Alarm schlagen muss. **Nimm nun die Drahtbrücke auf dem Breadboard zwischen Pin 6 und 8 heraus und lege sie beiseite.** Wenn alles stimmt, sollte die LED jetzt alle paar Sekunden mehrmals blinken – das Signal, dass es Zeit für die Gießkanne ist.

Jetzt kannst du den Pflanzenwächter – wenn du möchtest –zunächst wieder herausnehmen, in seine Box stecken und wieder neben die Pflanze setzen.

**Gieße anschließend die Pflanze – die LED hört nun auf zu blinken, bis die Erde zu trocken geworden und es Zeit für einen Schluck Wasser ist.**

# Der Code des Pflanzenwächters

Der Code besteht im Prinzip aus folgenden Teilen:

- Den Sensor kalibrieren
- Die Feuchtigkeit messen
- Alarm geben, wenn die Erde zu trocken ist
- Den ATtiny85 „schlafen legen“

Zu Beginn müssen allerdings zwei Bibliotheken eingebunden und ein paar Pins definiert werden:

## Die benötigten Bibliotheken

Du brauchst für den Code zwei Bibliotheken, die bereits vorinstalliert sind:

```
#include <avr/sleep.h>
#include <avr/wdt.h>
```

Mit der Bibliothek **avr/sleep.h** kannst du deinen ATtiny85 in den Schlafmodus versetzen, was dafür sorgt, dass sein Stromverbrauch erheblich sinkt. Da der Pflanzenwächter mit einer Batterie betrieben wird, ist dieser Schlafmodus besonders wichtig. Erfahre hier mehr über diese Bibliothek (Englisch).

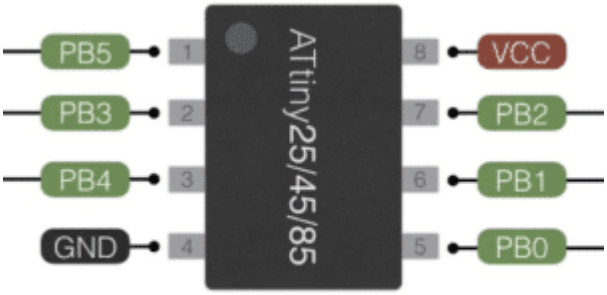
Die Bibliothek **avr/wdt.h** ist ein sogenannter Watchdog, den du – vereinfacht gesagt – dazu verwendest, deinen ATtiny85 nach einer bestimmten Zeit wieder aufzuwecken. In diesem Beitrag erfährst du hierüber mehr.

## Die Pins definieren

Insgesamt musst du vier Pins definieren, die dein ATtiny85 verwendet:

```
#define calibrationPin 1
#define sensorPin 2
#define sensorValuePin 3
#define ledPin 4
```

**Beachte hierbei, dass die Nummern der Pins nicht mit den Pin-Nummern übereinstimmen, die du vorhin gegen den Uhrzeigersinn durchgezählt hast.** Möchtest du die Pins deines Chips im Code ansteuern, benötigst du andere Nummern, die du ebenfalls in diesem Schema findest:




http://github.com/SpenceKonde/ATTinyCore

Die Nummern für deinen Code beginnen mit **PB**. Wie du siehst, entspricht zum Beispiel der erste Pin neben der Vertiefung dem „Code-Pin“ **PB5**. Im Code lässt du die Buchstaben jedoch weg und schreibst nur 5. Insgesamt stehen dir nach Abzug von Plus (VCC) und Minus (GND) noch 6 Pins zur freien Verfügung.

Beim Pflanzenwächter hängt die LED zum Beispiel neben dem „Beinchen“ Nr. 3 des ATtiny85 – im Code steuerst du sie jedoch mit dem Pin 4 an:

```
#define ledPin 4
```

> *In den folgenden Erläuterungen sind ab jetzt immer die Pin-Nummern des Codes gemeint.*

Im Setup des Sketchs legst du nun noch fest, ob es sich um Eingänge (INPUT) oder Ausgänge (OUTPUT) handelt:

```
void setup() {
  pinMode(calibrationPin, INPUT);
  pinMode(sensorPin, OUTPUT);
  pinMode(sensorValuePin, INPUT);
  pinMode(ledPin, OUTPUT);
}
```

## Die Kalibrierung des Pflanzenwächters

Kommen wir zum Loop des Sketchs. Hier fragst du zunächst ab, ob die die Drahtbrücke zur Kalibrierung des Sensors eingesteckt ist. Diese befindet sich ja zwischen Plus (VCC) und Pin 1 – **das heißt, dein ATtiny85 misst an diesem Pin einen Strom (HIGH), wenn die Drahtbrücke angeschlossen ist.**

Ist das der Fall, wird im Loop nur der Code für die Kalibrierung ausgeführt:

```
if (digitalRead(calibrationPin) == HIGH) {
  digitalWrite(sensorPin, HIGH);
  if (digitalRead(sensorValuePin) == 1) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Zunächst wird der Sensor angeschaltet, in dem der Pin für seine Stromzufuhr (sensorPin) auf HIGH gesetzt wird. In einer zweiten Abfrage wird die LED eingeschaltet, wenn der Sensor einen Messwert von 1 übergibt (an den Pin sensorValuePin). Ist die Erde feucht, wird die LED ausgeschaltet.

**Was trocken und was feucht bedeutet bestimmst du selbst durch Drehen des Potis auf der Platine des Feuchtigkeitssensors.** Wie gesagt, kalibrierst du deinen Pflanzenwächter am einfachsten, wenn du ihn in trockene Erde steckst und den Poti so drehst, dass die LED gerade noch leuchtet. Wenn du dann gießt, geht sie aus – bis die Erde wieder so trocken wie bei der Kalibrierung ist.

## Die Feuchtigkeit messen

Kommen wir zur eigentlichen Messung der Feuchtigkeit. **Hier kann es nur zwei Zustände geben: Die Erde ist zu trocken oder Die Erde ist noch feucht genug.**

Zunächst wird der Sensor für 100 Millisekunden eingeschaltet:

```
digitalWrite(sensorPin, HIGH);
delay(100);
```

Wenn der Sensor dann eine 1 misst und weitergibt, bedeutet das, dass die Erde zu trocken ist. Dann wird zunächst der Sensor wieder ausgeschaltet, um sofort wieder Strom zu sparen. Dafür blinkt jedoch die LED 10 mal auf und ab.

Zuletzt wird der Watch Dog scharfgemacht, sodass der ATtiny85 schlafen geht und nach 8 Sekunden wieder aufwacht und die LED wieder blinken lässt – sofern du die Pflanze in der Zwischenzeit nicht gegossen hast.

```
if (digitalRead(sensorValuePin) == 1) {

  digitalWrite(sensorPin, LOW);

  for (byte i = 0; i < 10; i++) {
    digitalWrite(ledPin, HIGH);
    delay(200);
    digitalWrite(ledPin, LOW);
    delay(200);
  }

  myWatchdogEnable (0b100001);
}
```

Fall du das jedoch gemacht hast, schlägt der Pflanzenwächter natürlich keinen Alarm mehr. Er misst jetzt, dass die Erde feucht genug ist und übergibt eine 0 an den entsprechenden Pin. **In dem Fall geht sowohl die LED als auch der Sensor aus und der Watchdog schaltet deinen ATtiny85 für 30 Minuten in den Schlafmodus**:

```
else {
  digitalWrite(ledPin, LOW);
  digitalWrite(sensorPin, LOW);
```

```
  for (byte j = 0; j <= 225; j++){
    myWatchdogEnable (0b100001);
  }
}
```

Zum Einsatz kommt hier die Funktion **myWatchdogEnable()**, die du ebenfalls im Sketch findest. Dieser Funktion gibst du das Byte **0b100001** mit, das den Timer auf 8 Sekunden stellt. Der For-Loop sorgt dafür, dass dieser 8-Sekunden-Timer 225 Mal ausgeführt wird – was 30 Minuten Schlaf ergibt.

Bleibt nur noch eine Frage: Wie funktioniert der Watchdog? Das ist eine auch für Fortgeschrittene keine so einfache Sache. Aber in diesem Beitrag erfährst du mehr über die grundlegende Funktionsweise.

# Der vollständige Sketch

Hier nun der vollständige Sketch – falls du noch nicht weißt, wie du ihn auf deinen ATtiny85 hochlädst, erfährst du hier, wie das funktioniert.

```
/*
   Code für den ATtiny85
   https://polluxlabs.net
   Copyright: CC BY-NC-ND
*/

#include <avr/sleep.h>
#include <avr/wdt.h>

#define calibrationPin 1
#define sensorPin 2
#define sensorValuePin 3
#define ledPin 4

void myWatchdogEnable(const byte interval)
{
  MCUSR = 0;                          // reset various flags
```

```
  WDTCR |= 0b00011000;               // see docs, set WDCE,
WDE
   WDTCR =  0b01000000 | interval;    // set WDIE, and
appropriate delay

  wdt_reset();
  set_sleep_mode (SLEEP_MODE_PWR_DOWN);
  sleep_mode();            // now goes to Sleep and waits for
the interrupt
}

void setup() {
  pinMode(calibrationPin, INPUT);
  pinMode(sensorPin, OUTPUT);
  pinMode(sensorValuePin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {

   //wenn am calibrationPin Strom anliegt, kann der Sensor
kalibriert werden
  if (digitalRead(calibrationPin) == HIGH) {
    digitalWrite(sensorPin, HIGH);
    if (digitalRead(sensorValuePin) == 1) { // 1 = trocken
      digitalWrite(ledPin, HIGH);
    } else {
      digitalWrite(ledPin, LOW);
    }

    //wenn kein Strom anliegt, geht er in den Messmodus
    //Code, wenn die Erde zu trocken ist:
  } else {
    digitalWrite(sensorPin, HIGH);
    delay(100);
    if (digitalRead(sensorValuePin) == 1) {

      digitalWrite(sensorPin, LOW);

      for (byte i = 0; i < 10; i++) {
        digitalWrite(ledPin, HIGH);
```

```
        delay(200);
        digitalWrite(ledPin, LOW);
        delay(200);
      }

      myWatchdogEnable (0b100001);

    }
    //Code, wenn die Erde noch feucht genug ist:
    else {
      digitalWrite(ledPin, LOW);
      digitalWrite(sensorPin, LOW);

      for (byte j = 0; j <= 225; j++) {
          myWatchdogEnable (0b100001);  // 8 seconds --
verlängern auf viel Zeit
      }
    }
  }
}

ISR(WDT_vect)
{
  wdt_disable();  // disable watchdog
}
```