1 Einfacher Stromkreis

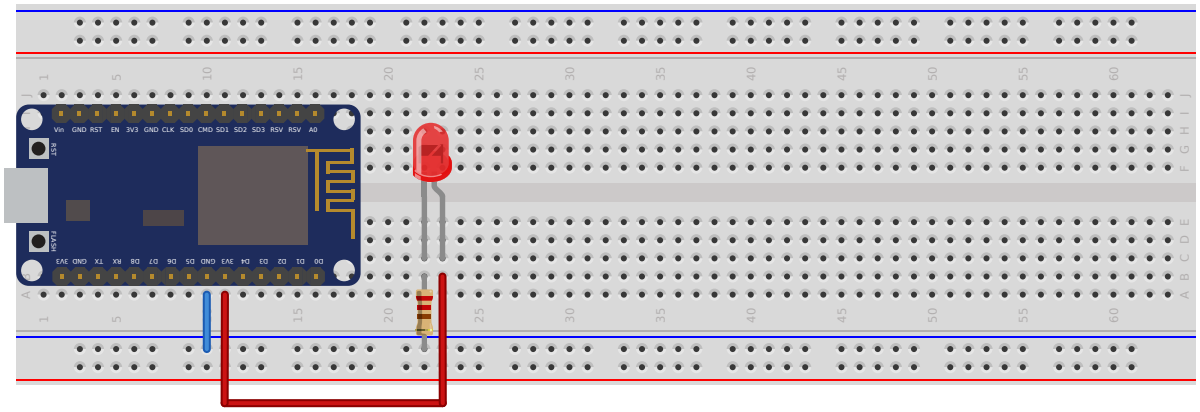
In der ersten Aufgabe wollen wir einen einfachen Stromkreis aufbauen, um zu verstehen, wie Strom fließt. Dabei folgt er immer diesen Grundregeln:

- Strom fließt von Plus nach Minus
- Strom nimmt **immer** den Weg mit dem kleinsten Widerstand.
- Wenn er ohne einen Verbraucher direkt vom Plus- zum Minuspol der Stromquelle fließt, kann die Stromquelle kaputtgehen (Kurzschluss)

Material

- 1* Breadboard
- 1* ESP8266
- 1* LED
- 1* Vorwiderstand passen zu der LED (frag bei Unsicherheiten auf jeden Fall nach)
- 2* Kabel

Aufbau



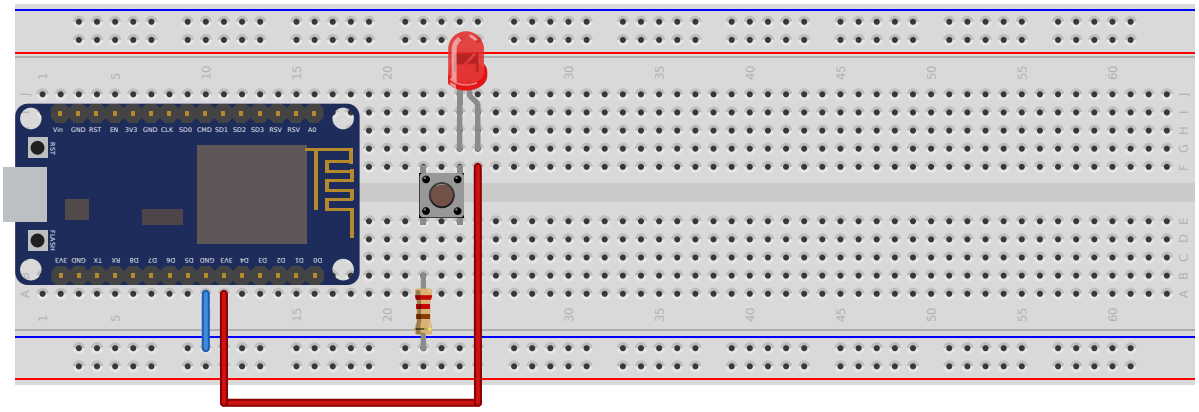
2 Stromkreis mit Taster unterbrechen

Nun wollen wir unsere eben gebaute Schaltung um einen Taster erweitern, womit wir die LED an- und ausschalten können.

Stromkeis mit Taster

- 1* Breadboard
- 1* ESP
- 1* LED
- 1* Taster
- 1* Vorwiderstand passend zu der LED
- 2* Kabel

Aufbau



So nun sollte deine LED immer dann aufleuchten, wenn du auf den Taster drückst.

3 Blinklicht

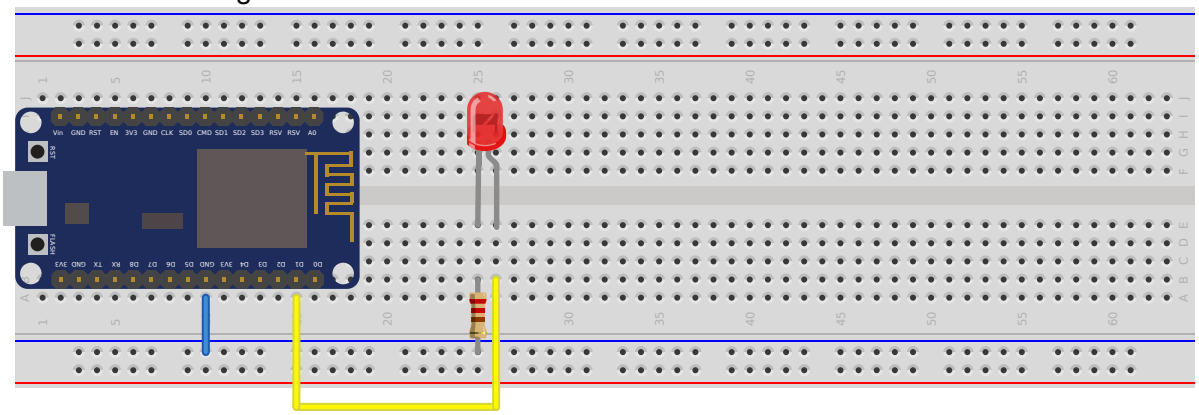
In diesem Versuch werden wir das erste Mal anfangen zu programmieren. Wir werden erst einmal, um die gängigsten Befehle kennenzulernen, eine LED zum Blinken bringen.

Material

- 1* Breadboard
- 1* ESP
- 1* LED
- 1* Vorwiderstand passend zu der LED
- 2* Kabel

Aufbau

Du kannst wieder ganz einfach nach der Aufbauskizze aufbauen.



Programierung

Wir programmieren den ESP in der Programmiersprache Micropython. Dies ist eine Abwandlung der bekannten Programmiersprache Python, welche auch auf Mikrocontrollern läuft.

```
1  #Modul fuer Zeit(bei uns, um das Programm anzuhalten) einbinden
2  import time
3  #Module, um an die Hardware ranzukommen einbinden
4  import machine
5  #Die Zeit zwischen an und aus festlegen
6  frequenz = 1
7  #Nummer unseres Led-Pins festlegen
8  pinNumber = 5
9  #Led-Pin als Ausgangs-Pin definieren
10 led = machine.Pin(pinNumber, machine.Pin.OUT)
11 #Dauerschleife
12 while(True):
13         #Schalte die LED an
14         led.off()
15         #Warte 1 sec
16         time.sleep(frequenz)
17         #Schalte die LED aus
18         led.on()
19         #Warte 1 sec
20         time.sleep(frequenz)
21
```

In Z. 2 und 4 binden wir zwei Module ein, die uns mehr Funktionen bieten. In Z. 6 und 8 erstellen wir drei Variablen für den LED-Pin, die Blinkfrequenz und die eigentliche LED. In Z. 10 beginnen wir eine Dauerschleife. In Z. 12 und 14 schalten wir die LED an bzw. aus. In Z. 16 und 18 warten wir jeweils eine Sekunde, bevor wir die LED wieder an- bzw. ausschalten.

4 Wechselblinklicht

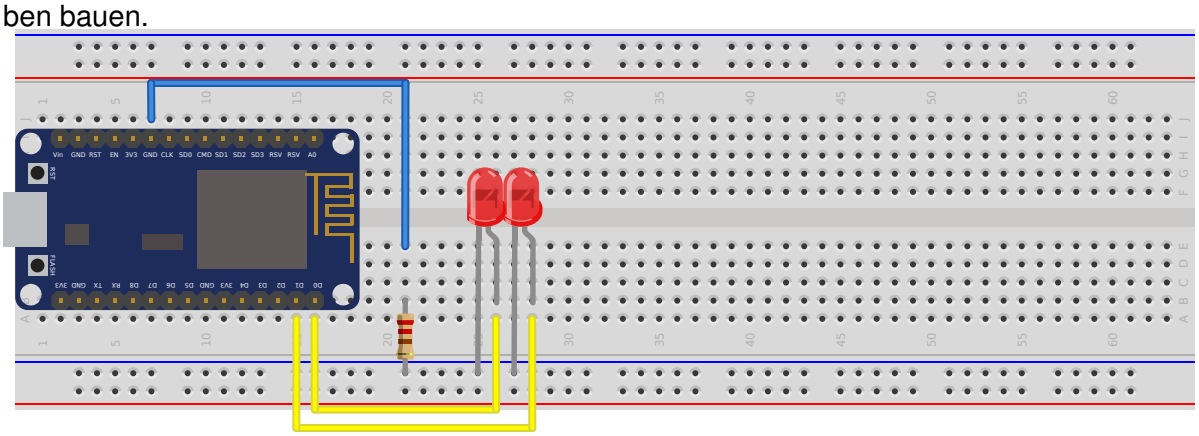
Wir erweitern unseren vorherigen Versuch um eine LED und lassen diese im Wechsel blinken.

Material

- 1* Breadboard
- 1* ESP
- 2* LED
- 1* Vorwiderstand passend zu den LEDs
- 3* Kabel

Aufbau

Die Aufbauskizze unterscheidet sich nur gering vom voherigen Versuch. Einfach eine zweite LED daneben bauen.



Programmierung

Das vorherige Programm lässt sich ganz einfach erweitern um die LEDs im Wechsel blinken zu lassen.

```
1  #Module einbinden
2  import time
3  import machine
4  #Frequenz und Pinnummern festlegen
5  frequenz = 1
6  led1Pin = 16
7  led2Pin = 5
8  #Pins definieren
9  led1 = machine.Pin(led1Pin, machine.Pin.OUT)
10 led2 = machine.Pin(led2Pin, machine.Pin.OUT)
11 #Dauerschleife
12 while (True):
```

```
13          led1.on()
14          time.sleep(frequenz)
15          led1.off()
16          led2.on()
17          time.sleep(frequenz)
18          led2.off()
19
```

Die LEDs werden im Wechsel an und wieder aus geschaltet.

5 Lauflicht

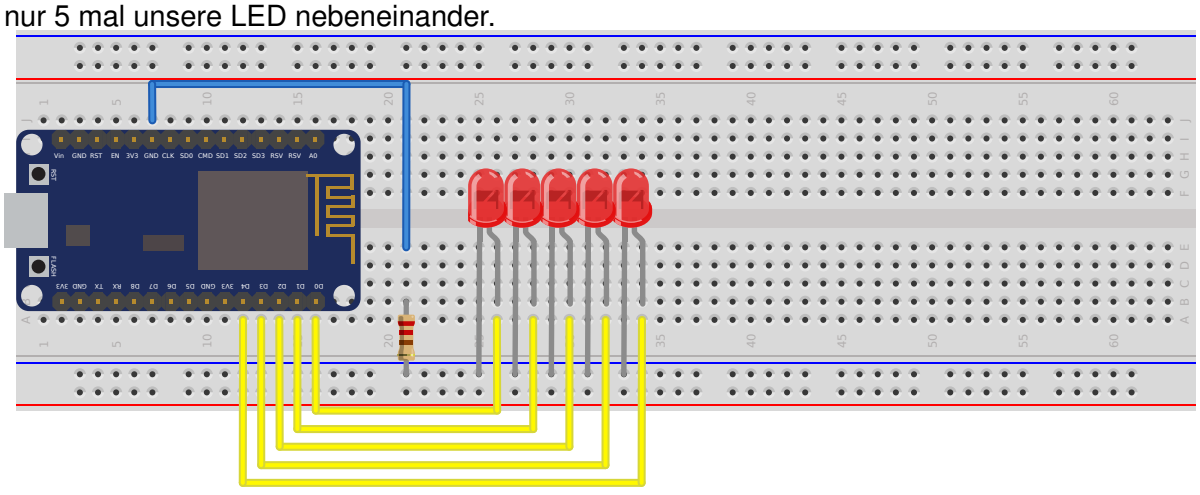
Dieses Projekt ähnelt sich dem vorherigen Projekt. Es kommen auch keine neuen Bauteile, Schaltungen oder Befehle dazu, aber du kannst hierbei versuchen möglichst viel aus dem Gedächtnis zu programmieren, denn Wiederholung schadet dem Verständnis nie.

Material

- 1* Breadboard
- 1* ESP
- 5* LED
- 1* Widerstand passend zu den LEDs
- 6* Kabel

Aufbau

Die Aufbauskizze mag zwar zuerst vewirrend scheinen, allerdings ist es im Grunde genommen einfach nur 5 mal unsere LED nebeneinander.



Programmierung

Die Programmierung ist jetzt auch kein Hexenwerk. Du lässt da praktisch auch nur LEDs nacheinander blinken.

```
#Module einbinden
import time
import machine
#Frequenz und Pinnummern festlegen
frequenz = 1
led1Pin = 16
led2Pin = 5
led3Pin = 4
led4Pin = 0
led5Pin = 2
#Pins definieren
led1 = machine.Pin(led1Pin, machine.Pin.OUT)
led2 = machine.Pin(led2Pin, machine.Pin.OUT)
led3 = machine.Pin(led3Pin, machine.Pin.OUT)
led4 = machine.Pin(led4Pin, machine.Pin.OUT)
led5 = machine.Pin(led5Pin, machine.Pin.OUT)
#Dauerschleife
while (True):
        led1.on()
        time.sleep(frequenz)
        led1.off()
        led2.on()
        time.sleep(frequenz)
        led2.off()
        led3.on()
        time.sleep(frequenz)
        led3.off()
        led4.on()
        time.sleep(frequenz)
        led4.off()
        led5.on()
        time.sleep(frequenz)
        led5.off()

```

In der Dauerschleife schalten wir immer zuerst die vorherige LED aus, dann die aktuelle an und warten dann die festgelegte Zeit, bis wir wieder die vorherige ausschalten u.s.w.

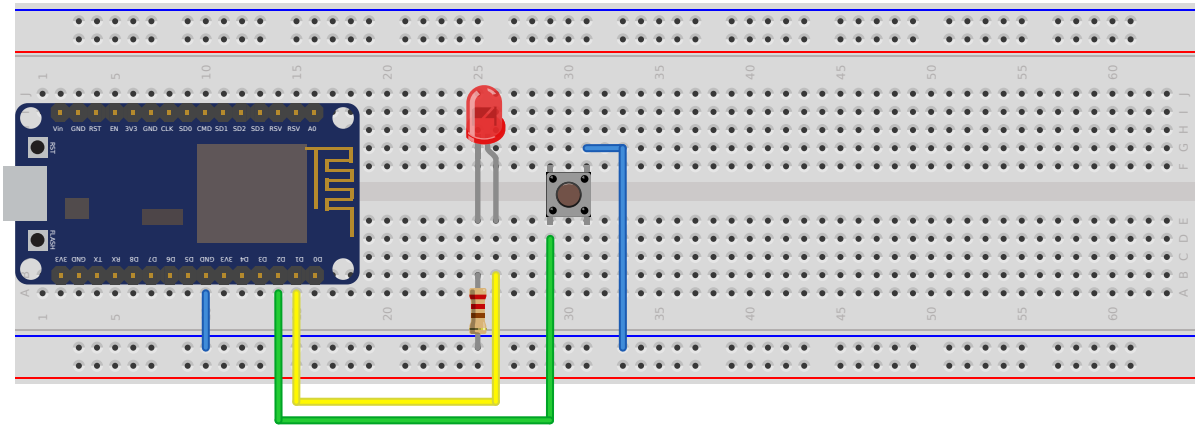
6 Taster auslesen

Nun möchten wir auch auf äußere Einflüsse reagieren können. Um damit ganz einfach anfangen zu können, nutzen wir eine einfache LED, um diese bei Tastendruck anzuschalten.

Material

- 1* ESP
- 1* Taster
- 1* LED
- 1* Widerstand passend zur LED
- 7* Kabel
- 1* Breadboard

Aufbau



Programmierung

Für dieses Projekt benötigen wir zwei oder drei neue Befehle.
Den ersten benutzen wir um den Zustand des Tasters abzufragen, er heißt `pin.value()`.
Der zweite Befehl heißt `if` und ist eine Bedingung. Er wird z.B. eingesetzt, wenn der Button gedrückt wird. Wenn diese Bedingung erfüllt ist, wird der dazugehörige Code ausgeführt.
Der letzte und dritte Befehl lautet `else` und hat sehr viel mit dem zweiten Befehl zu tun. Wenn nämlich die Bedingung nicht erfüllt wird, wird(wenn vorhanden) das ansonsten also `else` ausgeführt.

Nun muss nur noch der Code mit den Drei neuen Befehlen geschrieben werden:

```
# LED wird an Pin 0 angeschlossen.
ledPin = 5
# Der Taster wird an Pin 2 angeschlossen.
tasterPin = 4

# Die Pin Objekte erstellen
led = machine.Pin(ledPin, machine.Pin.OUT)
# Wir teilen dem Taster einen internen Pullup-Widerstand zu, um einen genauen Messwert zu erhalten.
taster = machine.Pin(tasterPin, machine.Pin.IN, machine.Pin.PULL_UP)

while True:
        tasterStatus = taster.value()
        if(tasterStatus == False):
                led.off()
        else:
                led.on()


```

7 Ultraschallsensor

Als nächsten wollen wir mit einem Sensor den Abstand zu einem Gegenstand ermitteln.

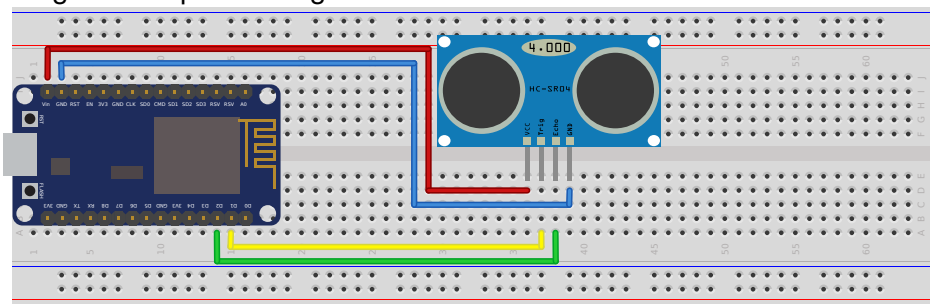
Material

- 1* Breadboard
- 1* ESP8266
- 1* Abstandssensor
- 3* Kabel

Aufbau

Hier kommt ein neues Bauteil zum Einsatz, der Ultraschallsensor. Der Ultraschallsensor besteht aus einem Ultraschall-Sender und einem Ultraschall-Empfänger. An diesen wird, wie in der Skizze gezeigt über die beiden äußeren Pins eine Spannung angelegt (Achtung, der Ultraschallsensor benötigt 5V um zu funktioniern) Der Trigger-Pin ist mit dem Sender, der Echo-Pin mit dem Empfänger verbunden. Der Microcontroller an dem Trig-Pin 10 Microsekunden den Wert 1 anlegt. Sendet das Modul einen Ultraschallimpuls aus. Dieser Impuls wird vom Sensor empfangen. Die Zeit zwischen dem Senden des Sigals und dem Empfang wird gemessen. Je näher der Gegenstand an dem Sensor ist, umso kleiner

ist die Zeitspanne. Der Sensor gibt danach auf den Echo-Pin eine hohe Spannung aus. Die Dauer der Ausgabe entspricht der gemessenen Zeit.



Programmierung

Nun geht es an die Programmierung. Hier lernen wir einen neuen Befehl kennen. Mit dem Befehl `machine.time_pulse_us()` kann die Dauer eines Impulses gemessen werden.

```
1  #Module einbinden
2  import time
3  import machine
4
5  echo = machine.Pin(4, machine.Pin.IN)
6  trigger = machine.Pin(5, machine.Pin.OUT)
7
8  while True:
9          trigger.off()
10         time.sleep_us(5)
11         trigger.on()
12         time.sleep_us(10)
13         trigger.off()
14
15         duration = machine.time_pulse_us(echo, 1)
16
17         distance = (duration/2) / 29.1
18
19         print("Distanz in cm: " + str(distance))
20
21         time.sleep(0.5)
22
```

8 Farbenwechsel

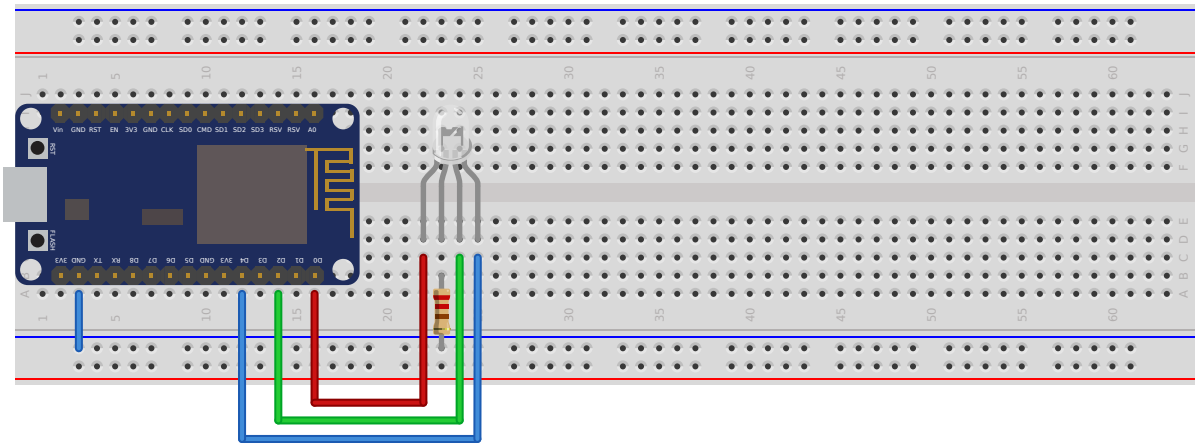
Als nächstes schauen wir uns die RGB-LED an. Diese kann man sich vorstellen, wie drei normale LEDs in einem Gehäuse. Es gibt drei Beinchen die man bei Plus anschließt und eins das man bei Minus (GND) anschließt. Nun kann man die "drei LEDs" getrennt ansteuen, und durch anschalten der roten und der blauen LED lila erzeugen. Man kann die drei LEDs auch dimmen und so noch mehr Farbmischungen

erzeugen. Das wird der Befehl sein, den wir in diesem Aufbau lernen. Dieser Befehl heißt `duty()`. Dafür brauchts du ein PWM Objekt. Dieses kannst du mit `pwmObjekt = machine.PWM(machine.Pin(pinNummer), freq=1000)` anlegen. Du benutzt ihn mit folgenden Argumenten `pwmObjekt.duty(pwmWert)`. Den Versuchsaufbau kannst du der Abbildung entnehmen.

Material

- 1* RGB-Led
- 1* ESP8266
- 1* Breadboard
- 1* Wiederstand 200
- 4* Kabel

Aufbau



Programmierung

So nun geht es an den Code. Wir wollen zunächst die Farbwerte der drei Farben erst einmal in jeweils einer Variable speichern. Und dann möchten wir jeden Pin der LED/Farbe mit dem neu erlernten Befehl einzeln ansteuern.

```
1 #Die ganzen Werte der drei Variablen kann man sich selbst zwischen 0 und 1023 aussuchen
2 #Den Roten Teil der LED auf volle Leistung schalten
3 rotWert = 1023
4 #Den Gruenen Teil der LED auf aus schalten
5 gruenWert = 0
6 #Den Blauen Teil der LED auf volle Leistung schalten
7 blauWert = 1023
8 #Die Anschlusspins der Drei LED-anschluesse definieren
9 #Welche Belegung welche Farbe ist, muesst ihr selbst ausprobieren
```

```
10  rotPin = 16
11  gruenPin = 4
12  blauPin = 2
13
14  rot = machine.PWM(machine.Pin(rotPin), freq=1000)
15  gruen = machine.PWM(machine.Pin(gruenPin), freq=1000)
16  blau = machine.PWM(machine.Pin(blauPin), freq=1000)
17
18  while True:
19          #Den drei LEDs den Wert zuweisen und entsprechend schalten
20          rot.duty(rotWert)
21          gruen.duty(gruenWert)
22          blau.duty(blauWert)
23
```

So, nun könnt ihr coole Muster oder eine Diskoleuchte bauen.

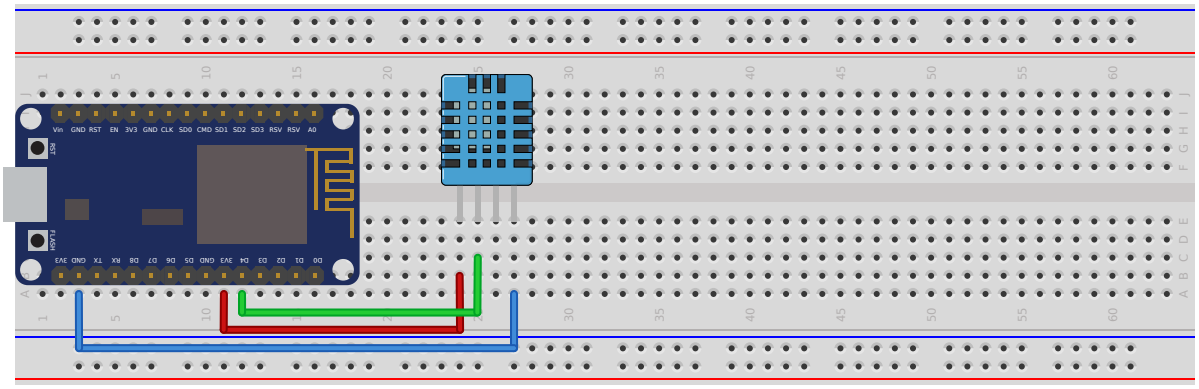
9 Temperatur- und Luftfeuchtigkeitsmessung

In diesem Versuch lernen wir den DHT11-Sensor kennen. Mit diesem kann man die Temperatur und die Luftfeuchtigkeit messen. Dieser Sensor hat vier Beinchen. Von vorne betrachtet wird das linke Beinchen mit dem 3V3-Pin verbunden und das rechte mit dem GND-Pin. Das zweite Beinchen von links wird an ein GPIO-Pin angeschlossen. Das zweite Beinchen von rechts wird nicht verbunden.

Material

- 1* ESP8266
- 1* Breadboard
- 1* DHT11
- 3* Kabel

Aufbau



Programmierung

So nun geht es an den Code. Dieser ist hier sehr übersichtlich, da du alles kennst, was du dafür brauchst. Das einzig neue ist die Bibliothek DHT. Diese brauchst du um den Sensor ansteuern zu können.

```
1  from machine import Pin
2  import dht
3  import time
4
5  #Sensor initialisieren
6  d = dht.DHT11(Pin(2))
7
8  while True:
9          d.measure()
10         temp = d.temperature()
11         hum = d.humidity()
12
13         print("Temperatur: "+str(temp)+"°C Luftfeuchtigkeit: "+str(hum) + "%")
14         time.sleep(2)
15
```

So, nun könnt ihr eine kleine Wetterstation bauen.