

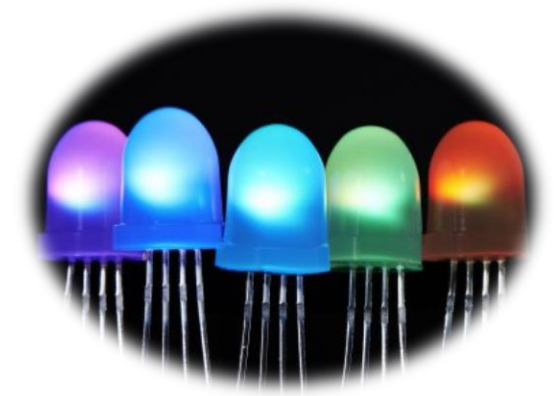
# Smartes Stimmungslicht

smile

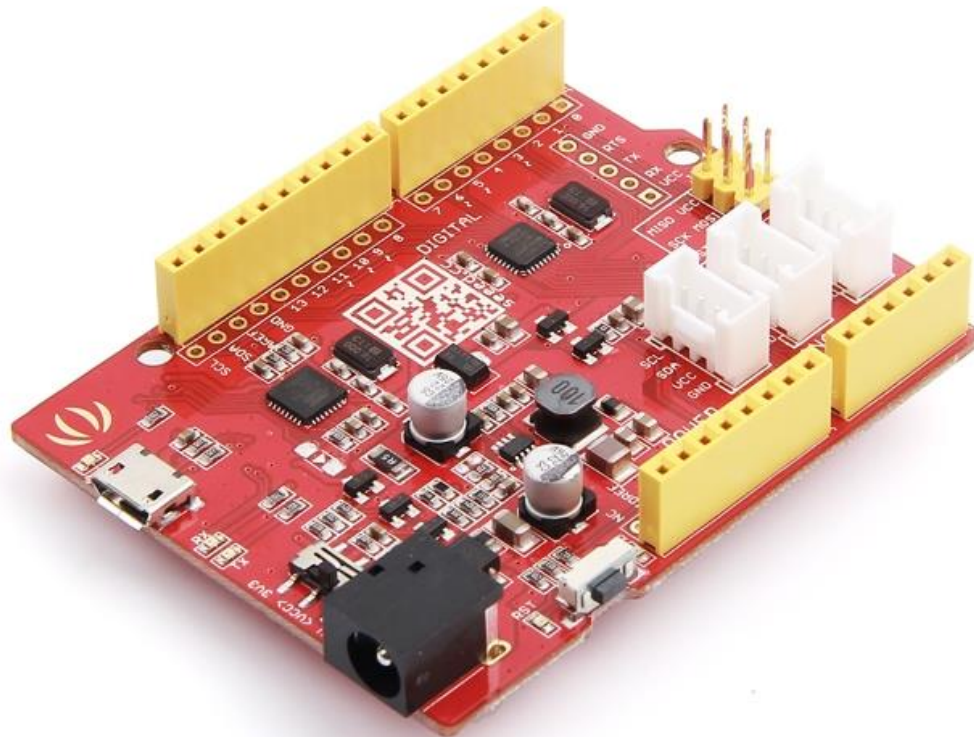
---

IT

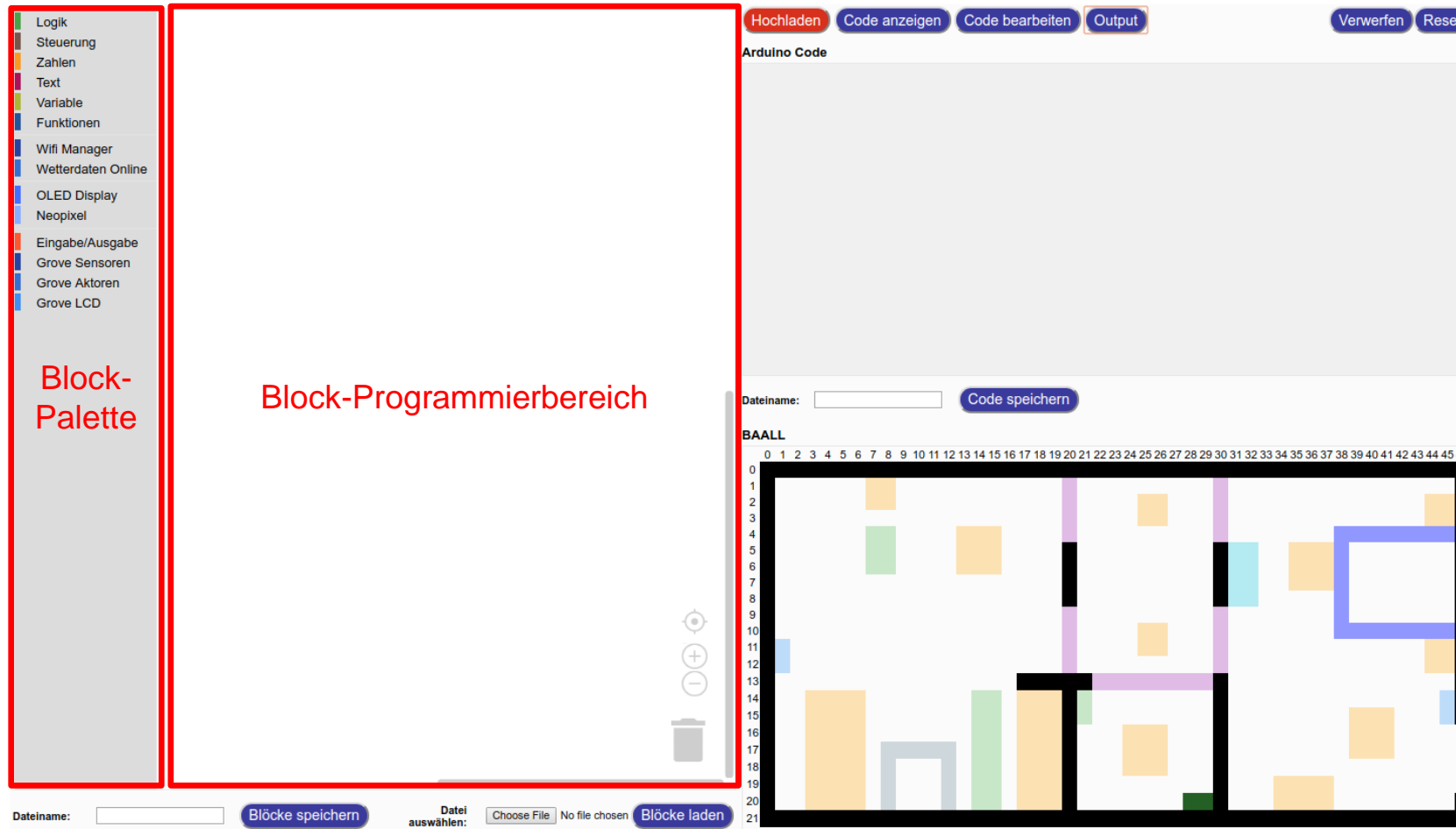
SMART · FUTURE · ME



# Hands-On Arduino



# Die Entwicklungsumgebung: BEEISM



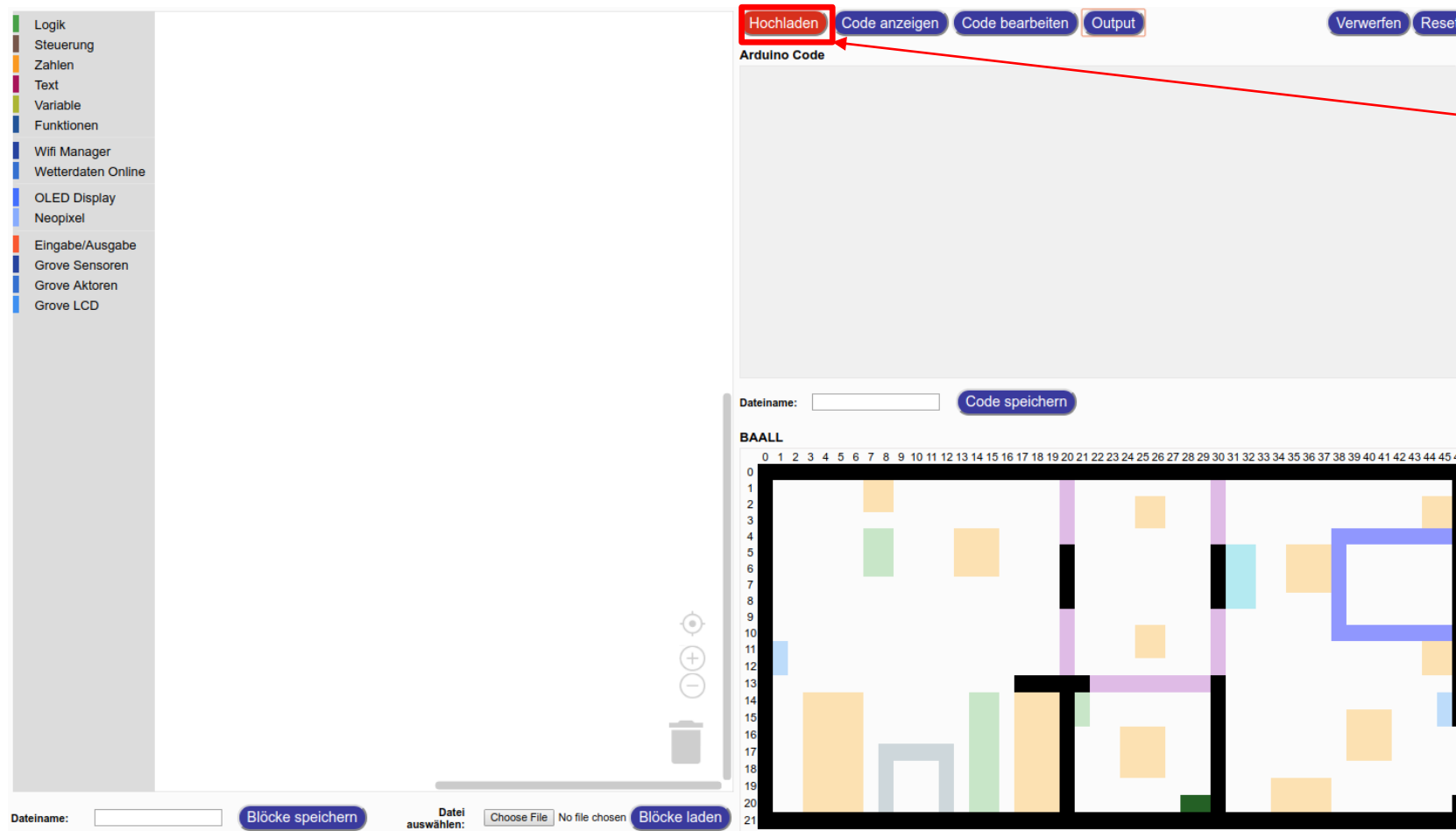
In der **Blockpalette**  
und im  
**Programmierbereich**  
arbeiten wir und  
stellen unser  
Programm aus  
verschiedenen  
Blöcken zusammen.

# Die Entwicklungsumgebung: BEEISM



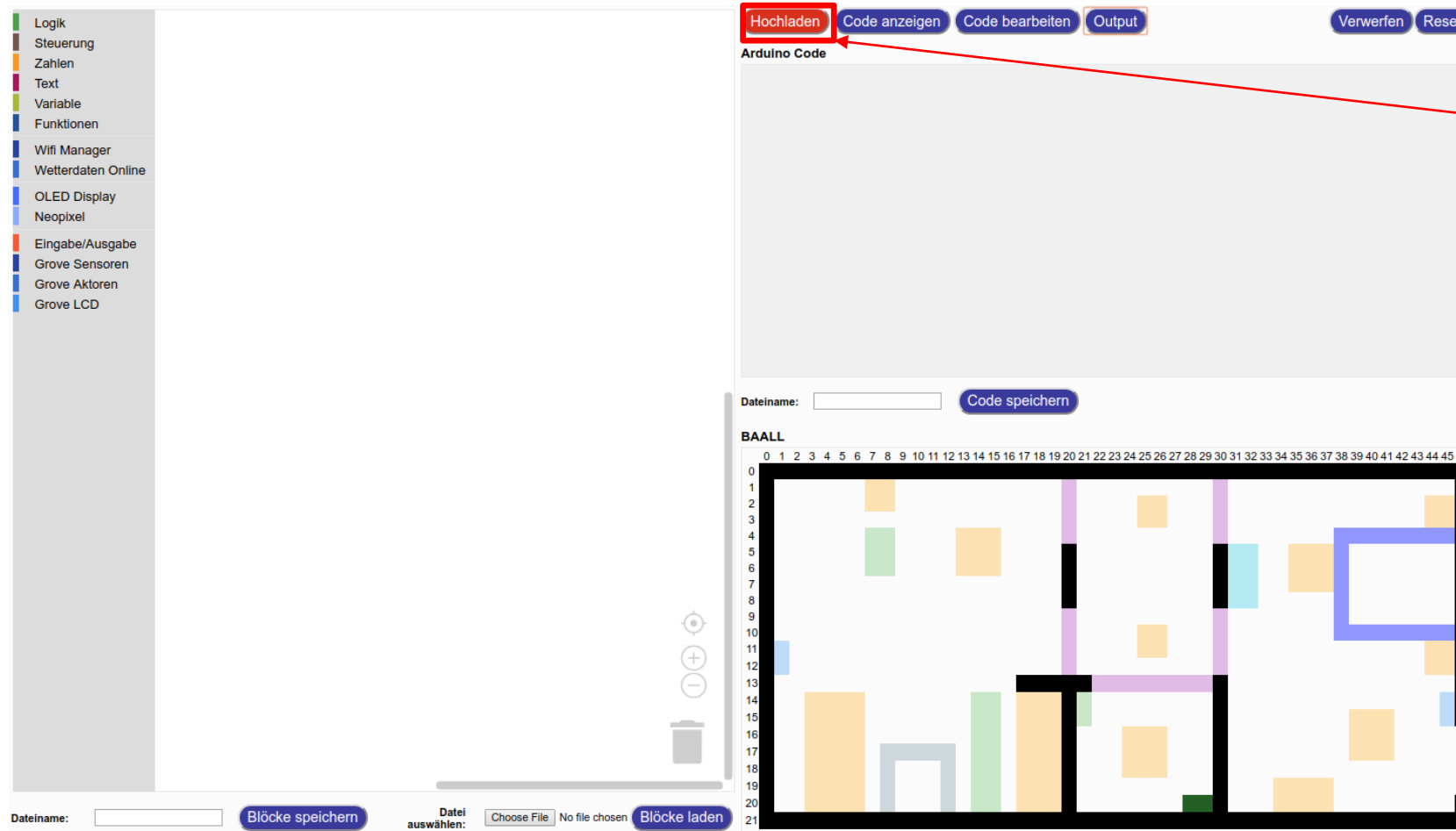
Im **Code Bereich** sehen wir den Code, der aus den Blöcken, die wir im Block-Programmierbereich zusammengestellt haben, generiert wird.

# Die Entwicklungsumgebung: BEEISM



Mit einem Klick auf **Hochladen** wird das Programm auf den Controller übertragen. Dies kann ein bisschen dauern. Also Geduld!

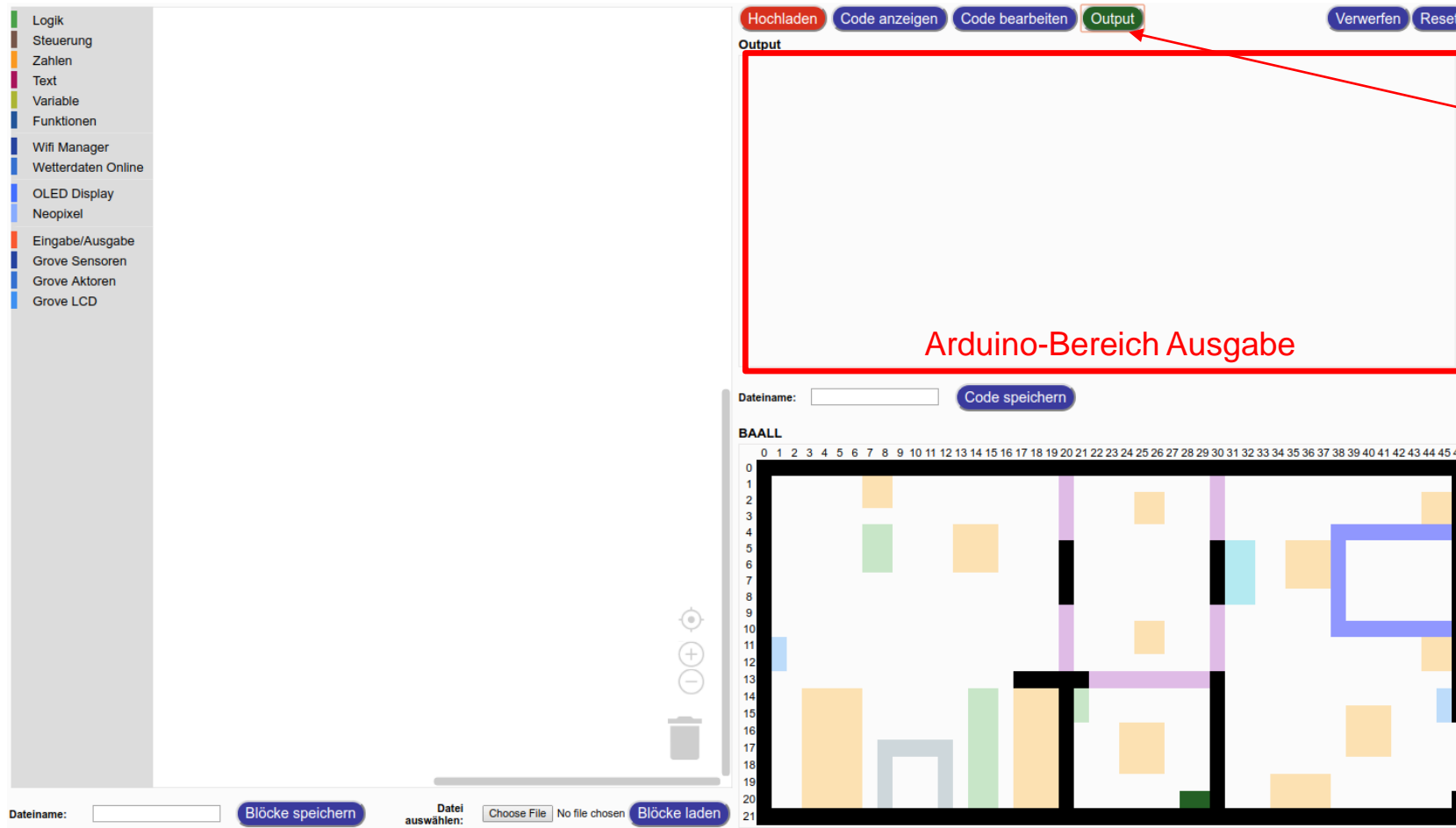
# Die Entwicklungsumgebung: BEEISM



Wenn das Übertragen erfolgreich war, erscheint „Das Hochladen war erfolgreich“.

Wenn das Übertragen nicht erfolgreich war, erscheint „Upload war nicht erfolgreich“. Dann muss herausgefunden werden woran es lag.

# Die Entwicklungsumgebung: BEEISM



Im **Ausgabe Bereich** können wir sehen was unser Programm gerade macht. Also ob es uns z. B. einen Fehler ausgibt.

# Die Entwicklungsumgebung: BEESM

The screenshot displays the BEESM development environment. On the left is a vertical sidebar with a category list: Logik, Steuerung, Zahlen, Text, Variable, Funktionen, Wifi Manager, Wetterdaten Online, OLED Display, Neopixel, Eingabe/Ausgabe, Grove Sensoren, Grove Aktoren, and Grove LCD. The main workspace is divided into two primary sections. The top section, titled 'Arduino Code', contains a large text editor and a toolbar with buttons: 'Hochladen', 'Code anzeigen', 'Code bearbeiten', 'Output', 'Verwerfen', and 'Reset'. The bottom section features a grid-based block editor with a coordinate system (0-46 on both axes) and a toolbar with 'Blöcke speichern', 'Datei auswählen', 'Choose File', 'No file chosen', and 'Blöcke laden'. A red rectangular box highlights a portion of the block editor, and the text 'Vorschau BAALL' is overlaid in red within this area.



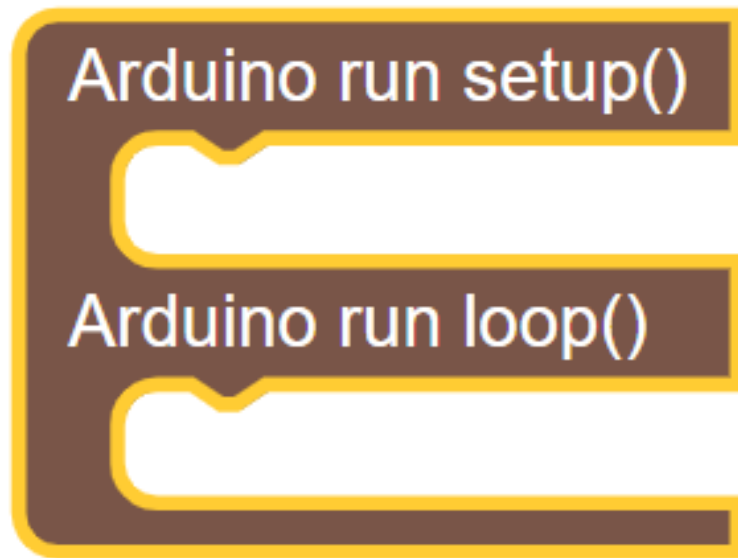
# Die Entwicklungsumgebung: BEESM



```
guest@smile-linux1: ~  
guest@smile-linux1:~$
```

./start.sh

# Arduino Programmstruktur



- Setup:
  - Wird **einmal** zu Beginn des Programms ausgeführt.
- Loop:
  - Wird nach dem Setup als **Endlosschleife** ausgeführt

# Der Loop

## Arduino Code

```
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel strip_D3 = Adafruit_NeoPixel(1 , D3 , NEO_GRB + NEO_KHZ800);

void setup()
{
  strip_D3.begin();

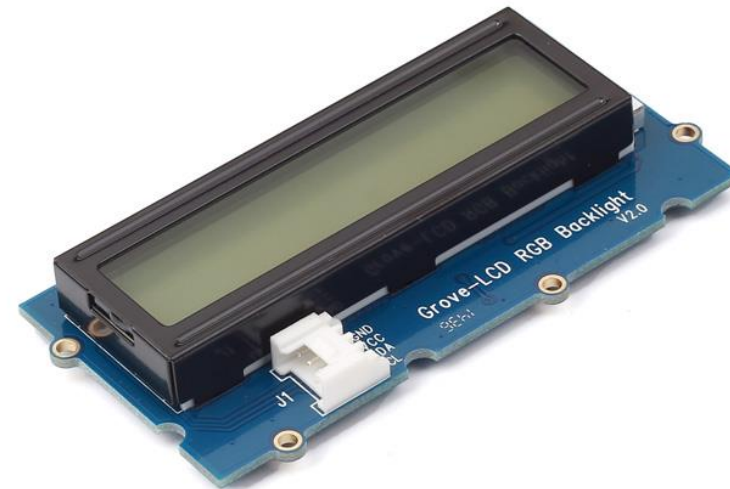
  strip_D3.setBrightness(150);

  strip_D3.show();
}


void loop()
{
  strip_D3.setPixelColor(0,200,0,200);
  strip_D3.show();
  delay(1000);
  strip_D3.setPixelColor(0,0,0,0);
  strip_D3.show();
  delay(1000);
}
```

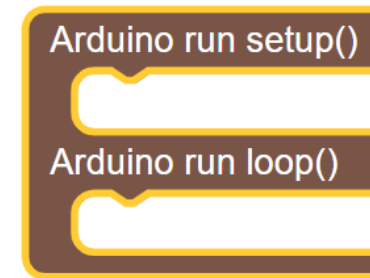
Loop()

# Das RGB LC Display



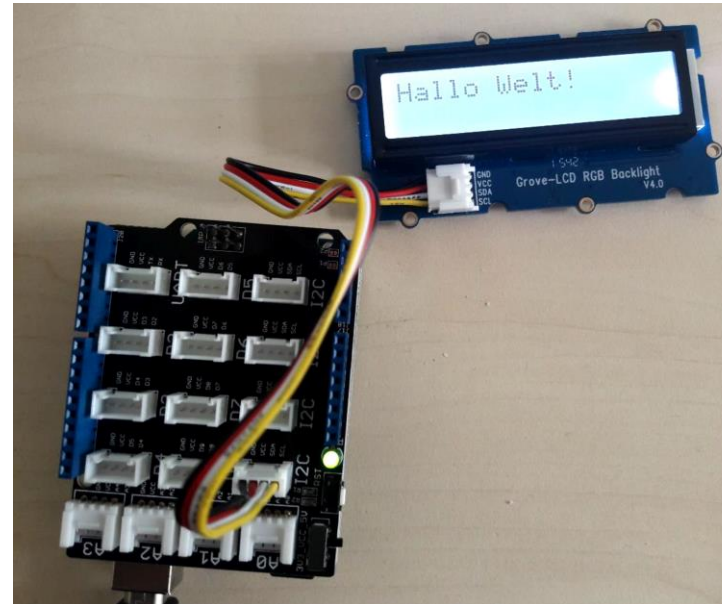
# Ein erstes Programm

- Schreibe ein Programm, das „Hallo Welt“ auf dem Display ausgibt.
- Beachte dabei folgende Schritte:
  - Ziehe den Block für die Arduino Programmstruktur in den Programmierbereich
  - Ziehe die Anweisungen für die Text-Ausgabe in den Setup Bereich. Die Befehle findet Ihr unter Grove LCD
  - Lade dein Programm in den Controller hoch
  - Der Upload dauert ein paar Sekunden. Warte bis die Bestätigung „Programm erfolgreich in den Controller hochgeladen“ erscheint. 



# Mein erstes Programm

Das Resultat sollte ungefähr so aussehen:



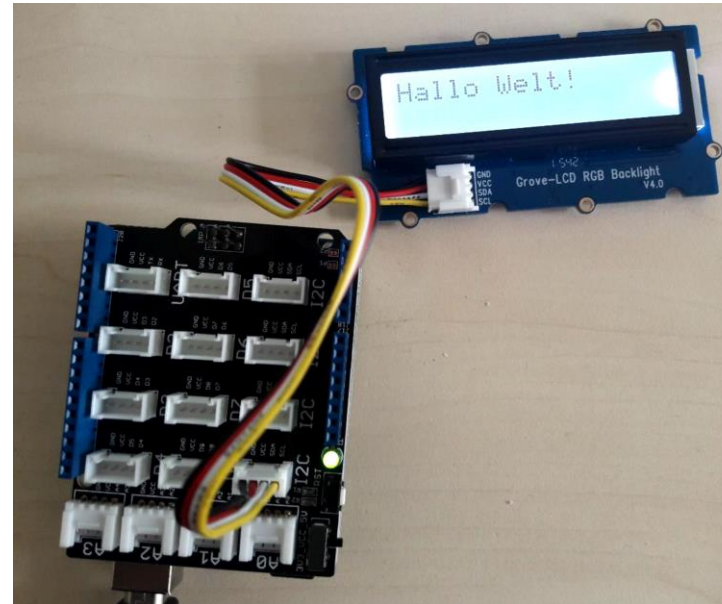
# Mein erstes Programm

Das Resultat sollte ungefähr so aussehen:

RGB LCD

Display anzeigen

“ Hallo Welt ”



# Übungen: RGB LCD


- Recherchiert:
  - Wofür steht LCD?
  - Wofür steht RGB?
- Ändert euer Programm
  - Ändert den Text, den Ihr ausgeben
  - Verändert die Position des Textes
  - Wieviel Text passt auf das Display?
- Speichert das Programm ab.





# Übungen 2: RGB LCD

- Ändert euer Programm

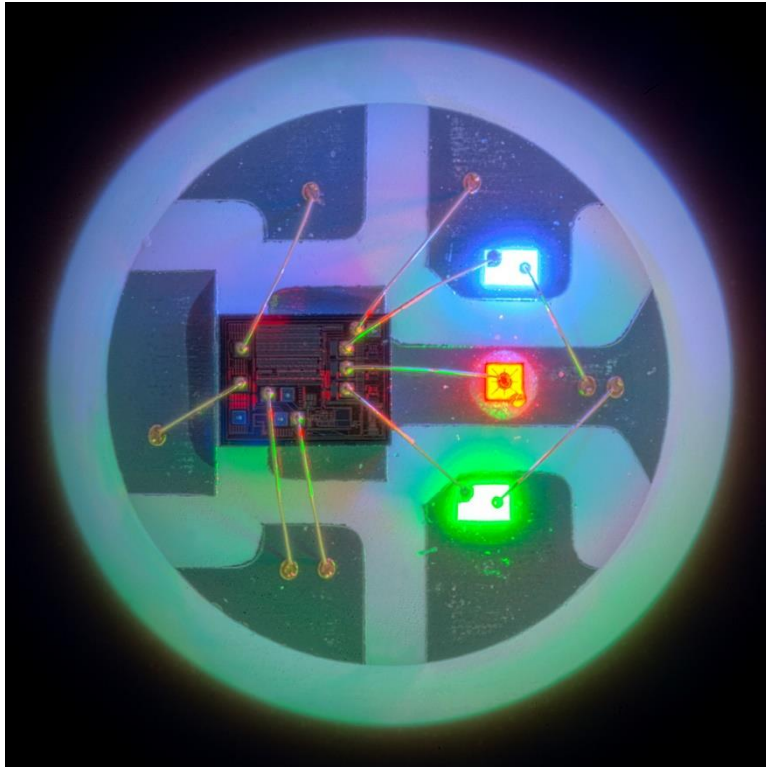
- Gebe den Text in der **Loop** aus und ziehe einen Block für eine Pause  zusätzlich in die Loop.
- Ändert die Hintergrundfarbe. Gebt dazu für die Felder rot, grün, und blau jeweils **Werte zwischen 0 und 255** an.

Tipps!  
1000 Millisekunden entspricht 1 Sekunde  
Den Block „warte“ findet ihr in der  
Blockpalette unter Steuerung

Frage 1: Was passiert, wenn Ihr den Text in der Loop ausgeben lasst? Warum?  \_

Frage 2: Was bedeutet die Zahl in dem „warte“ Block?  \_

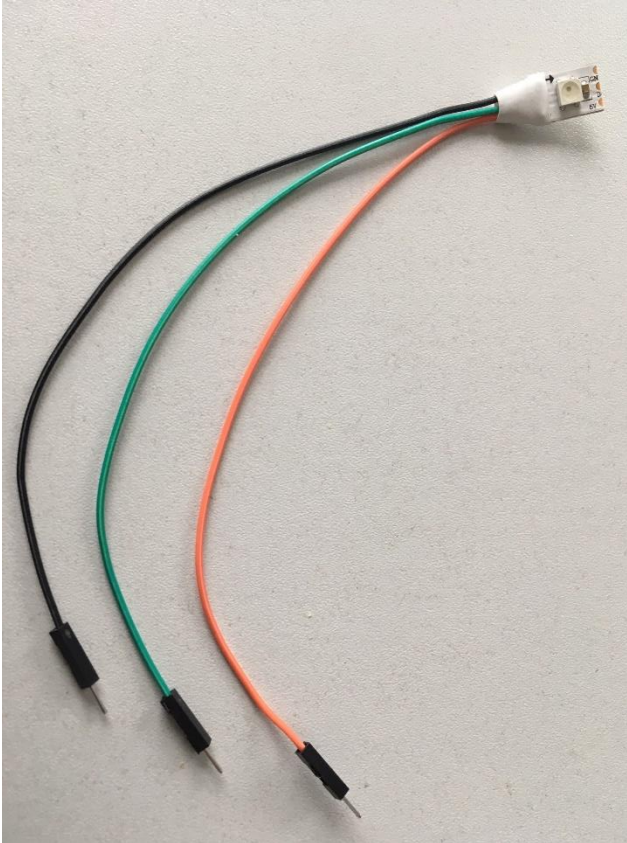
# RGB-LED



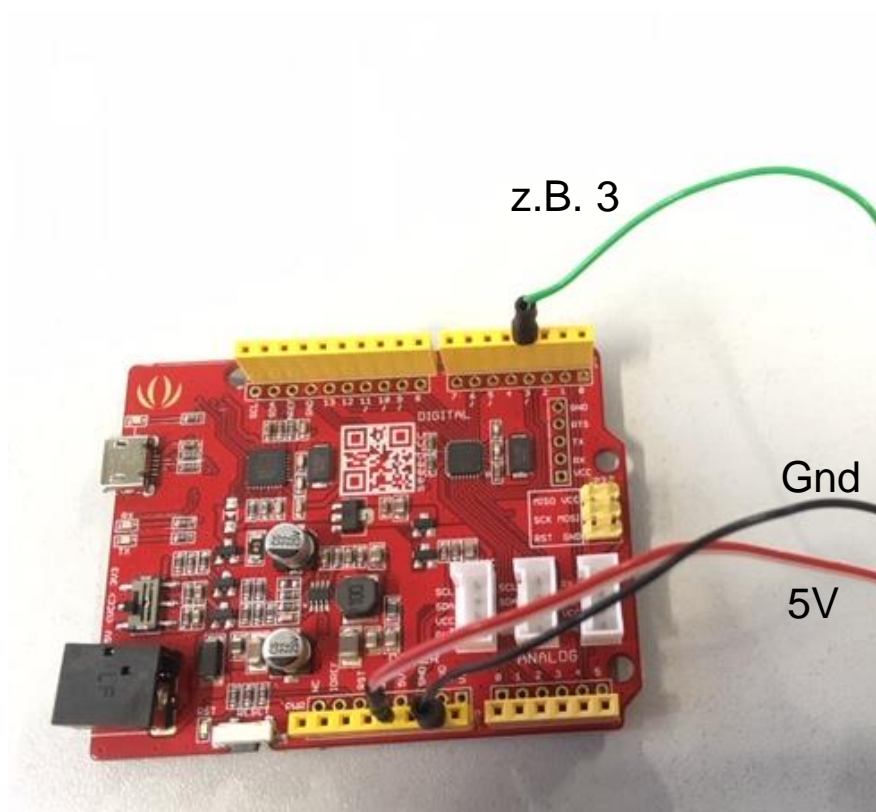
- Neopixel (→ „schlaue“ Pixel)
- Ein Lämpchen hat ein rotes, grünes und blaues Pixel, damit kann man 16,7 Mio. Farben mischen.



# RGB-LED-Anschließen



# RGB-LED-Anschließen



— + Strom (5V)

— - Masse, Ground (G, GND)

— Daten, Digital (1...x)

# RGB-LED-Neopixel

Neopixel anmelden pin# 1 Anzahl Pixel 0 Typ NEO\_GRB NEO\_KHZ800

Helligkeit setzen pin# 1 0

Farbe setzen pin# 1 Pixelnummer 0 rot 0 grün 0 blau 0

Wichtig! Bei allen Blöcken muss die **Pin** angegeben werden, an welchem wir die LED angeschlossen haben. Beim Arduino wird der Pin mit 1... 12 angegeben.

Neopixel anzeigen pin# 1

**Anmelden:** Wir definieren den **Pin**, an welchen wir die Datenleitung (gelbes, grünes oder blaues Kabel) unserer LED-Lämpchen angeschlossen haben und **wieviele LEDs** wir in Reihe geschaltet haben. Dies geschieht 1mal im Setup des Programms.

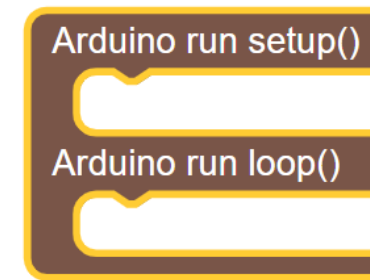
**Optional:** Wir können die **Helligkeit** der Lämpchen setzen. Ohne Angabe leuchten die Lämpchen mit voller Intensität, das ist gleich 255. Diese Angabe sollte nur einmal am Anfang des Programms im Setup gesetzt werden, wenn wir die Lämpchen insgesamt weniger hell haben wollen.

**Farbe setzen:** Hier sagen wir dem Programm, in welcher Farbe das LED-Lämpchen leuchten soll. Die Farbe wird aus den Werten für rot, grün und blau gemischt. Die Werte können jeweils zwischen 0 (aus) und 255 (höchste Intensität) liegen. **Pixelnummer** 0 bedeutet, dass wir das erste Lämpchen einschalten.

Die oben gesetzten Farben muss ich nun an die LEDs senden. Sonst passiert nichts. Diese Anweisung brauchen wir nach jeder Änderung der Lämpchen.

# Ein weiteres Programm

- Schreibe ein Programm, dass das Lämpchen in rot leuchten lässt.
- Beachte dabei folgende Schritte:
  - Schließt das LED Lämpchen an den Controller
  - Ziehe den Block für die Arduino Programmstruktur in den Programmierbereich
  - Wir müssen die LEDs im Setup **anmelden** und dann zum leuchten bringen. Die passenden Befehle findet Ihr unter Neopixel
  - Ladet euer Programm in den Controller hoch



# Übungen LED



# Was ist ein Programm?

- Habt Ihr da Ideen?



# Was ist ein Programm?

Habt Ihr da Ideen?

- Ein Programm ist eine Liste von Befehlen an einen Computer
- Wie ein Kochrezept oder eine Anleitung
- Die Befehle werden in der Reihenfolge abgearbeitet
- Oft sind Programmiersprachen aus simplen Befehlen aufgebaut, diese können aber kombiniert werden

# Wiederholung: Variablen

Programmiert einen Zähler:

- Wartet nach der Ausgabe der Variable für eine Sekunde
- Erhöht dann den Wert der Variable um eins
- Gebt die geänderte Variable aus
- **Wie seid ihr vorgegangen?**

# Wiederholung: Variablen

Programmiert einen Zähler:

- Wartet nach der Ausgabe der Variable für eine Sekunde
- Erhöht dann den Wert der Variable um eins
- Gebt die geänderte Variable aus
- Wie seid ihr vorgegangen?

Neu: Programmiert einen Countdown, der von 10 bis 0 runterzählt.

# Wiederholung: Variablen

Programmiert einen Zähler:

- Wartet nach der Ausgabe der Variable für eine Sekunde
- Erhöht dann den Wert der Variable um eins
- Gebt die geänderte Variable aus
- Wie seid ihr vorgegangen?

Neu: Programmiert einen Countdown, der von 10 bis 0 runterzählt.



# Wiederholung: RGB-LED

Lasst das LED-Lämpchen in einer beliebigen Farbe langsam angehen.

Diesen Effekt des langsamen Aus- oder Angehens nennt man **Fading**.

**Wie seid ihr vorgegangen?**

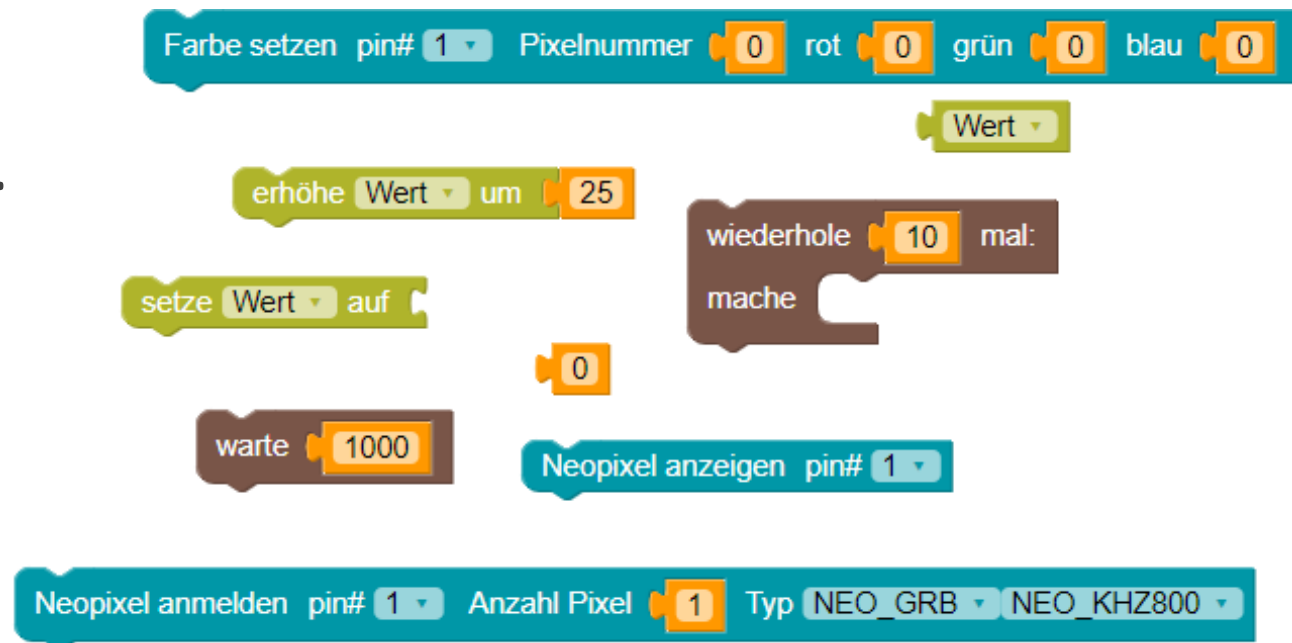
# Übung: RGB-LED

Lasst das LED-Lämpchen in einer beliebigen Farbe langsam angehen.

Diesen Effekt des langsamen Aus- oder Angehens nennt man **Fading**.

Benutzt nebenstehende  
Blöcke jeweils nur einmal.  
Wie geht ihr vor?

Speichert euer Programm!







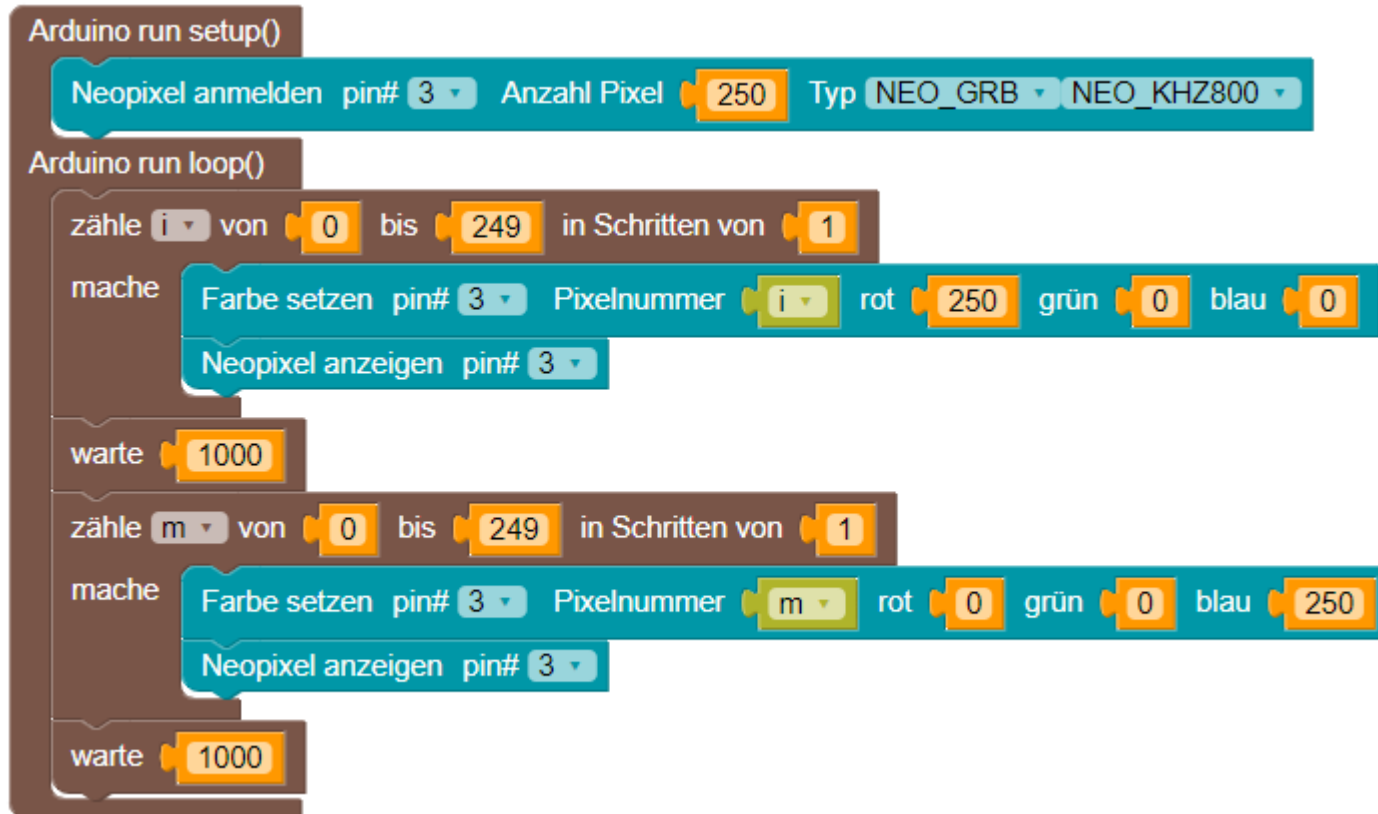
# Arbeitsblätter

For Schleife:

- LED Ketten
- LED Fading



# Schleifen



Was passiert hier?