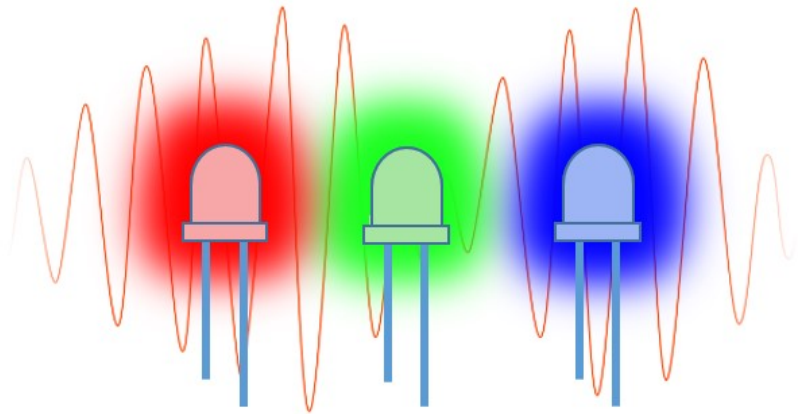


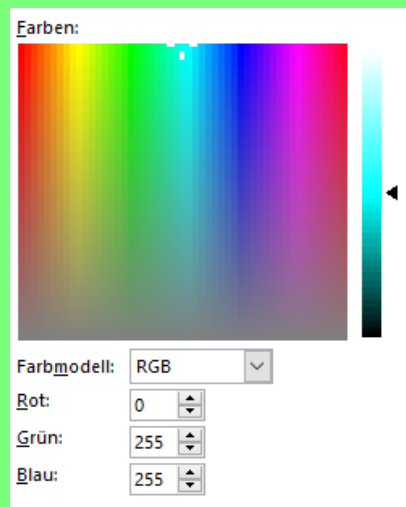
# 6



## Lösung: Erzeuge Farbmischung

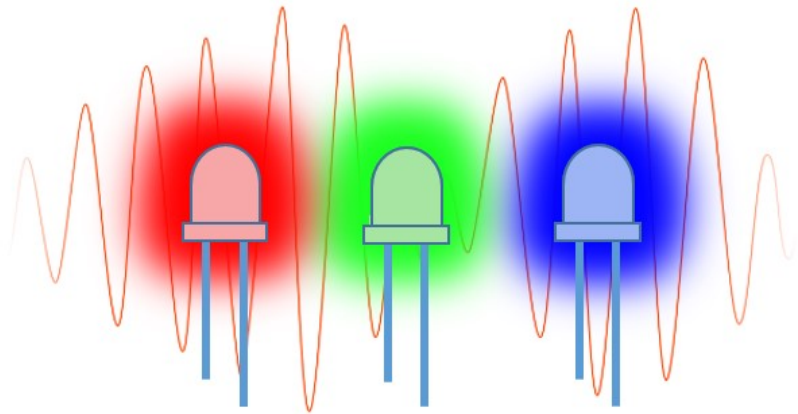
Der Color Picker wird in vielen bekannten Computerprogrammen verwendet und ist im Web auch als Onlinewerkzeug auffindbar.

Dieses RGB-Farbmischsystem nennt man Additive Farbmischung. Eine Eigenheit dieses Mischsystems ist, dass weisses Licht entsteht, wenn alle Grundfarben zusammen leuchten!



The screenshot shows the PGLU IDE interface with three code snippets in the 'Code' tab. Each snippet is a 'Hauptloop' (Main loop) set to '100'000 mal pro Sekunde' (100,000 times per second) and 'Blinkcode: kurz 1 lang 1' (Blink code: short 1 long 1). The snippets are as follows:

- Snippet 1:** LED 1 is set to 0%, LED 2 is set to 100%, and LED 3 is set to 100%. The resulting color is cyan.
- Snippet 2:** LED 1 is set to 100%, LED 2 is set to 0%, and LED 3 is set to 100%. The resulting color is magenta.
- Snippet 3:** LED 1 is set to 100%, LED 2 is set to 50%, and LED 3 is set to 0%. The resulting color is orange.



## Lösung: Blinke mit zufälligen Farben

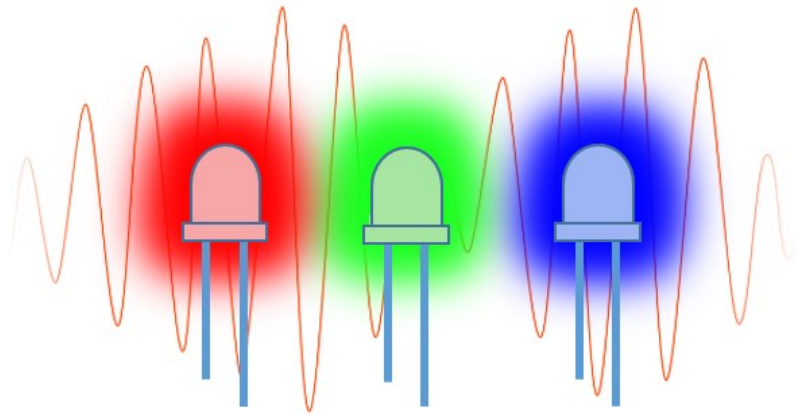
Mit diesem Programm können  $100 \times 100 \times 100 = 1$  Mio unterschiedliche Farbnuancen erzeugt werden!

Die Farbpixel auf deinem Handybildschirm bestehen ebenfalls aus 3 RGB-Lichtern und können sogar 16,7 Mio Farben darstellen! Grund ist, dass die Helligkeitsstufen normalerweise in 0 bis 255 Bit angegeben werden anstatt in Prozent.

The screenshot shows the PGLU programming environment with the following code block:

```
Hauptloop: 100'000 mal pro Sekunde | Blinkcode: kurz 1 | lang 1  
setze LED 1 auf ganzzahliger Zufallswert zwischen 0 bis 100 %  
setze LED 2 auf ganzzahliger Zufallswert zwischen 0 bis 100 %  
setze LED 3 auf ganzzahliger Zufallswert zwischen 0 bis 100 %  
pausiere 1000 ms
```

The interface includes a sidebar with categories: Aktion, Sensor, Motor, LED, Pixel, Logik, Frage, Ablauf, Zahl, Variable, Mathe, Struktur, Funktion, Loop, and Spezial. The main workspace shows the code block with a trash icon at the bottom right.



## Lösung: Blende die leuchtende LED1 langsam aus

Teste das Programm im Simulator und beobachte die Helligkeit von „Rot“ an Ausgang L1.

Beachte: wenn LED1=0% ist (LED ist dunkel), wird Variable „Rot“ nicht mehr um 1 vermindert.

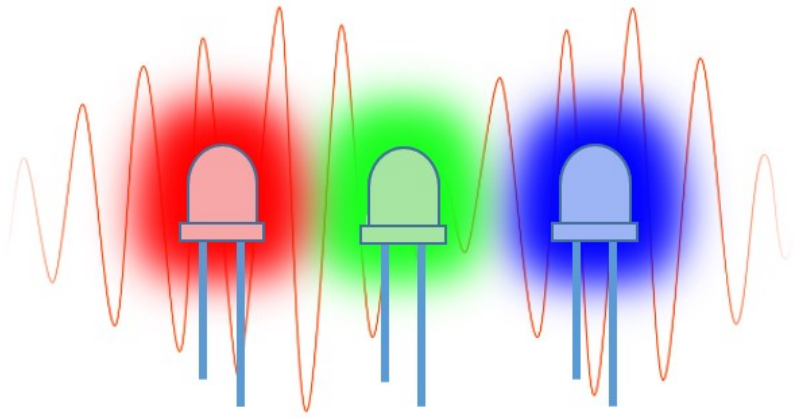
Dafür sorgt die Frage: „Ist Rot grösser als 0?“

Um das Ausblenden auf der Platine neu zu starten, drücke den kleinen Reset-Knopf direkt neben dem Prozessor.

The screenshot shows the PGLU simulator interface with the following components:

- Top Bar:** PGLU logo and tabs for Grafik, Code, Simulator, and Programme.
- Left Panel:** A vertical menu with categories: Aktion, Sensor, Motor, LED, Pixel, Logik, Frage, Ablauf, Zahl, Variable, Mathe, Struktur, Funktion, Loop, and Spezial.
- Simulation Area:** Four sensor inputs labeled Sensor 1 to Sensor 4, each with a switch and a digital display showing 0.
- Code Editor:**
  - Vor Hauptloop: 1x:** A block "schreibe rot = 100".
  - Hauptloop: 100'000 mal pro Sekunde:**
    - wenn:** "rot > 0".
    - schreibe:** "rot = rot - 1".
    - setze:** "LED 1 auf rot %".
    - pausiere:** "10 ms".
- Right Panel:** A slider for "Simulationsgeschwindigkeit".
- Bottom Bar:** Four LED status indicators labeled M1, M2, L1, L2, L3, L4. L1 is currently lit red and shows a value of 74.

9



## Lösung: Leuchte nach Musik und blende nach Peak aus

Teste das Programm im Simulator und klicke auf den Taster bei Sensor 4. Nach jedem Klick sollte der Pegel langsam von 100% auf 0% sinken.

Teste auch mit deiner Platine und justiere die Empfindlichkeit des Mikrofons mit einem feinen Schraubenzieher.

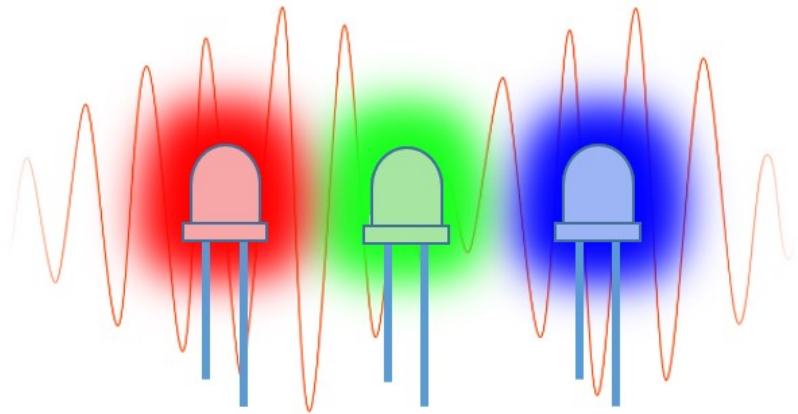
Übrigens: Wie schnell das Ausblenden geschieht, steuerst du mit der Länge der Programmpause. Anzahl Durchgänge bis Rot=0% x Pausenlänge, ergibt die Zeit für das vollständige Ausblenden.

The screenshot shows the PGLU simulator interface. The top navigation bar includes 'Grafik', 'Code', 'Simulator', and 'Programme'. The 'Simulator' tab is active, displaying four sensors (Sensor 1 to Sensor 4) with 'Schalten' (toggle), 'Regulieren' (slider), and 'Tasten' (button) controls. The 'Tasten' for Sensor 4 is highlighted. The main workspace shows a block-based program:

- Hauptloop:** 100'000 mal pro Sekunde
- Blinkcode:** kurz 1, lang 1
- Block 1:** wenn `lese Wert von Sensor 4 in % > 80` then `schreibe rot = 100`
- Block 2:** wenn `rot > 0` then `schreibe rot = rot - 1`
- Block 3:** setze LED 1 auf rot %
- Block 4:** pausiere 10 ms

The bottom status bar shows motor and LED states: M1 (0), M2 (0), L1 (62), L2 (0), L3 (0), L4 (0).

10



### Lösung: Leuchte zufallsfarbig nach Musik und blende nach Peak aus

Gratuliere, du hast das erweiterte Programm für deine Lichtorgel erfolgreich geschrieben. Experimentiere jetzt mit den einzelnen Werten und passe die Effekte deinem Musikgeschmack an!

In Varianten 1-6 findest du noch weitere Ideen für interessante Lichteffekte.

The screenshot shows a programming environment with a dark background and a sidebar on the left. The sidebar has categories: Aktion, Sensor, Motor, LED, Pixel, Logik, Frage, Ablauf, Zahl, Variable, Mathe, Struktur, Funktion, Loop, and Spezial. The main area shows a code block for a 'Hauptloop: 100'000 mal pro Sekunde' with a 'Blinkcode: kurz 1' and 'lang 1' dropdown. The code block contains the following steps:

- wenn `lese Wert von Sensor 4 in %` `>` `80`
- schreibe `rot` = `ganzzahliger Zufallswert zwischen 0 bis 100`
- schreibe `grün` = `ganzzahliger Zufallswert zwischen 0 bis 100`
- schreibe `blau` = `ganzzahliger Zufallswert zwischen 0 bis 100`
- pausiere `100` ms
- wenn `rot` `>` `0`
- schreibe `rot` = `rot` - `1`
- wenn `grün` `>` `0`
- schreibe `grün` = `grün` - `1`
- wenn `blau` `>` `10`
- schreibe `blau` = `blau` - `1`
- setze `LED 1` auf `rot` %
- setze `LED 2` auf `grün` %
- setze `LED 3` auf `blau` %
- pausiere `10` ms