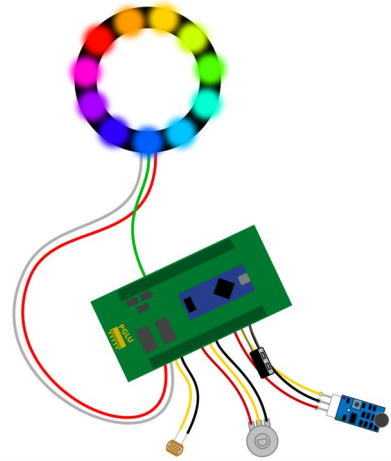


7



Rainbow still - Video 7

Ein Regenbogen besteht aus einem Farbverlauf über die gesamten 360° des Farbkreises. Bei einem Streifen mit 16 Pixeln ändert die Farbe zwischen zwei Pixeln also um $360^\circ / 16 = 22.5^\circ$

Ändere im Pixelblock die Option „LED-Streifen“ auf „Zwischenspeicher“

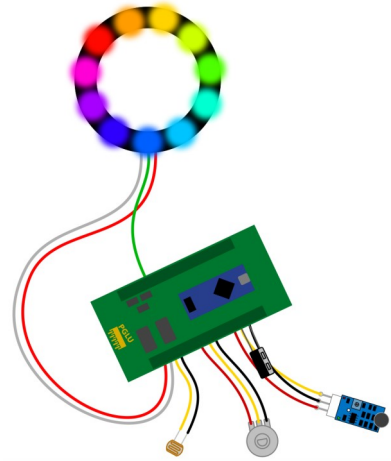
Das Bild des Regenbogens wird zuerst im Zwischenspeicher aufgebaut. Mit einer „zähle-von-bis-Schleife“ wird jedem einzelnen Pixel eine eigene Farbe gegeben. Erst wenn das ganze Bild fertig ist, wird es an den LED-Streifen gesendet

The screenshot shows the PGLU software interface with the following code blocks:

- definiere LED-Streifen Neopixel**: Anzahl Pixel: 16, Helligkeit total: 50 %
- Hauptloop: 100'000 mal pro Sekunde**: Blinkcode: kurz 1, lang 1
- zähle**: Pixel, von 1 bis 16, alle 1
- setze Zwischenspeicher**: ab Pixel: Pixel, Anzahl: 1, auf Farbe in °: Pixel * 22.5, Sättigung in %: 100, Helligkeit in %: 100
- sende Pixel aus Zwischenspeicher an LED-Streifen**

Annotations in the image:

- Yellow arrows point from the 'Zwischenspeicher' block to the 'Pixel' dropdown menu in the 'ab Pixel' field.
- A yellow arrow points from the 'Zwischenspeicher' block to the '22.5' value in the 'auf Farbe in °' field.
- A yellow arrow points from the 'Zwischenspeicher' block to the 'LED-Streifen' dropdown menu in the 'sende Pixel' block.
- A yellow arrow points to the 'LED-Streifen' dropdown menu with the text: "Ändere diese Option auf „Zwischenspeicher“"



Rainbow dynamisch - Video 8

Die zwei ineinander verschachtelten zähle-von-bis-Schleifen haben unterschiedliche Aufgaben:

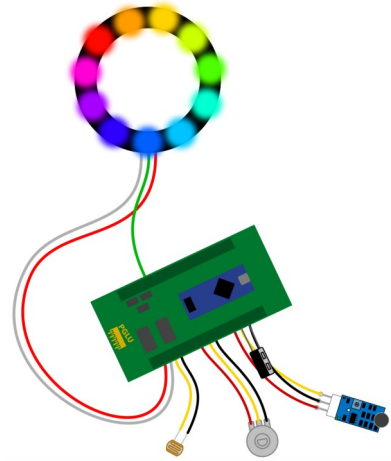
- Die innere zeichnet den Regenbogen wie in Schnipsel 7
- Die äussere Schleife versetzt das Farbmuster bei jedem Durchgang um 0.2° . Nach jedem Versatz wird das Bild an den Pixelstreifen gesendet

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

- definiere LED-Strip Neopixel**: Anzahl Pixel: 16, Helligkeit total: 50 %
- Hauptloop**: 100'000 mal pro Sekunde, Blinkcode: kurz 1, lang 1
- zähle**: Versatz Farbe, von 0 bis 359, alle 0.2
- zähle**: Pixel, von 1 bis 16, alle 1
- setze Zwischenspeicher ab Pixel**: Pixel
- Anzahl auf Farbe in °**: 1
- Sättigung in %**: 100
- Helligkeit in %**: 100
- sende Pixel aus Zwischenspeicher an LED-Strip**

The code uses nested loops to iterate over 360 color offsets (0 to 359) and 16 pixels (1 to 16). The inner loop sets the color for each pixel based on the current offset and a fixed saturation and brightness of 100%. The outer loop increments the color offset by 0.2 degrees for each iteration. The final step is to send the data from the memory buffer to the LED strip.

9



Musikpegel mit Hintergrund - Video 9

Dieses Snippet zeigt die Funktion des Zwischenspeichers anschaulich: Es können mehrere Bildebenen übereinandergelegt werden. Solche Ebenen können still oder bewegt sein.

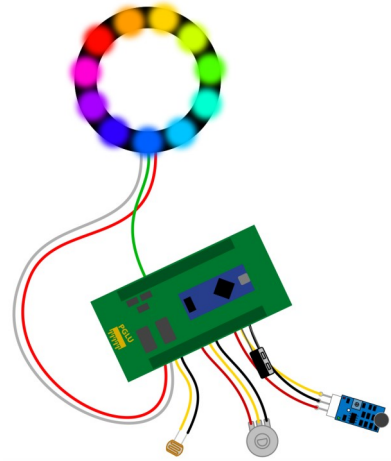
- Als erstes wird eine dunkle Ebene als Hintergrund gelegt
- Dann wird das ganze Magic Eye hellblau eingefärbt
- Über diese hellblaue Farbe wird der bewegte Musikpegel in feuerrot gelegt
- Erst jetzt wird dieses gesamte Bild ans Magic Eye gesendet!

Tipp: um dir das entstehende Bild gut vorstellen zu können, stelle dich in Gedanken unter den Loop und schaue nach oben auf die drei Ebenen. Wie auf dem Magic Eye siehst du so die feuerrote Ebene über der hellblauen und der schwarzen.

The screenshot shows the PGLU software interface with the following code blocks:

- definiere LED-Strip**: Anzahl Pixel: 16, Helligkeit total: 50 %
- Hauptloop: 100'000 mal pro Sekunde**: Blinkcode: kurz 1, lang 1
- setze LED-Strip im Zwischenspeicher auf dunkel**
- setze Zwischenspeicher ab Pixel 1**: Anzahl: 16, auf Farbe in °: 160, Sättigung in %: 100, Helligkeit in %: 100
- setze Zwischenspeicher ab Pixel 9**:
 - lese Wert von Sensor 4 in % * 0.08
- Anzahl**:
 - lese Wert von Sensor 4 in % * 0.16
- auf Farbe in °**: 10
- Sättigung in %**: 100
- Helligkeit in %**: 100
- sende Pixel aus Zwischenspeicher an LED-Strip**

10



Smooth Musikpegel - Video 10

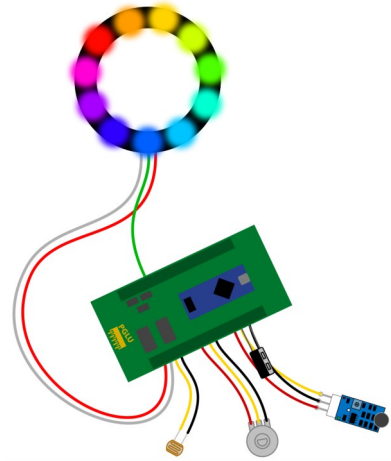
Eine Bildebene wird mit einem Musikpegel gelegt und an den Streifen gesendet. Dann bearbeitet eine „zähle-von-bis-Schleife“ jeden einzelnen Pixel im Zwischenspeicher und dimmt seine Helligkeit um 1%.

Pixel, die beim nächsten Messen des Musikpegels nicht mehr gesetzt werden, löschen dann nicht sofort ab, sondern blenden langsam aus.

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

- definiere LED-Strip Neopixel**: Anzahl Pixel Helligkeit total %
- Hauptloop: 100'000 mal pro Sekunde**: Blinkcode: kurz lang
- setze Zwischenspeicher ab Pixel**: (Sensor 4 in %)
- Anzahl**: (Sensor 4 in %)
- auf Farbe in °**:
- Sättigung in %**:
- Helligkeit in %**:
- sende Pixel aus Zwischenspeicher an LED-Strip**
- zähle Pixel von**: bis alle
- dimme Pixel**: im Zwischenspeicher um %

11



Spin - Video 11

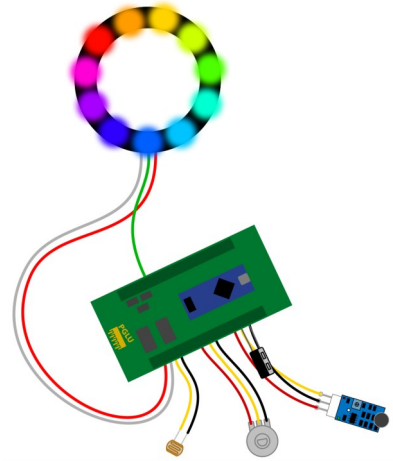
Wie in Schnipsel 10 wird auch hier das Magic Eye nie sofort auf dunkel gesetzt, sondern mit der „zähle-von-bis-Schleife“ langsam gedimmt. Eine zweite übergeordnete „zähle-von-bis-Schleife“ sorgt für das Kreisen des Pixels.

Die Funktion „Pausiere“ sorgt für eine Verlangsamung des Effektes und wird mit einem Poti reguliert.

The screenshot shows the PGLU software interface with the following code blocks:

- definiere LED-Strip Neopixel**: Anzahl Pixel: 16, Helligkeit total: 50 %
- Hauptloop: 100'000 mal pro Sekunde**: Blinkcode: kurz 1, lang 1
- zähle Pixel**: von 1 bis 16 alle 1
- setze Zwischenspeicher**: ab Pixel: Pixel
- Anzahl**: 1
- auf Farbe in °**: 20
- Sättigung in %**: 100
- Helligkeit in %**: 100
- sende Pixel aus Zwischenspeicher an LED-Strip**
- zähle Dimmer**: von 1 bis 16 alle 1
- dimme Pixel**: Dimmer: Dimmer, im Zwischenspeicher um %: 20
- pausiere**: lese Wert von: Sensor 3, in % ms

12



Sparkle Magic Eye & Matrix - Videos 12a & 12b

Ein Sternenregen mit vielen Zufallswerten und Dimmeffekt.

PGLU

Grafik Code Simulator Programme

Aktion

Sensor
Motor
LED
Pixel

Logik

Frage
Ablauf

Zahl

Variable
Mathe

Struktur

Funktion

Loop

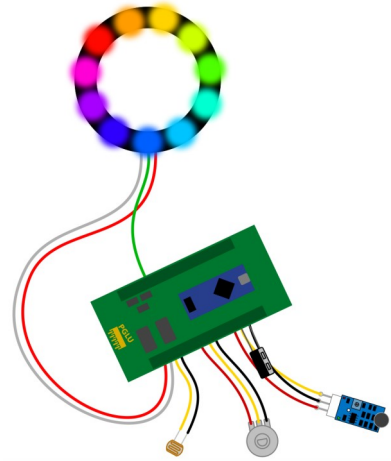
Spezial

```
definiere LED-Strip Neopixel Anzahl Pixel 16 Helligkeit total 30 %
```

Hauptloop: 100'000 mal pro Sekunde Blinkcode: kurz 1 lang 1

```
setze Zwischenspeicher ab Pixel ganzzahliger Zufallswert zwischen 1 bis 16  
Anzahl 1  
auf Farbe in ° ganzzahliger Zufallswert zwischen 0 bis 359  
Sättigung in % ganzzahliger Zufallswert zwischen 50 bis 100  
Helligkeit in % 100  
sende Pixel aus Zwischenspeicher an LED-Strip  
zähle Pixel von 1 bis 16 alle 1  
dimme Pixel Pixel im Zwischenspeicher um % 20  
pausiere 100 ms
```

13



Peak Matrix - Video 13

Eine zähle-von-bis-Schleife zählt von 1 bis 4 hoch, je nach Lautstärke der Musik. Diese Zahlen steuern die erste Spalte der Matrix, also Pixel 1-4.

Wenn du im Dokument „Neopixel Basics“ nachschaust, wie die Pixel auf der Matrix nummeriert sind, dann siehst du, dass dies in einer Schlangenlinie geschieht.

Mit den nachfolgenden Blöcken wird die erste Spalte nun auf die Spalten 2-4 kopiert. Zweimal seitenverkehrt und einmal gleich wie in Spalte 1.

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

- Aktion:** definiere LED-Strip Neopixel Anzahl Pixel Helligkeit total %
- Hauptloop:** 100'000 mal pro Sekunde Blinkcode: kurz lang
- zähle Pixel von 1 bis** **in %** **alle**
- setze Zwischenspeicher ab Pixel** **Anzahl** **auf Farbe in °** **Sättigung in %** **Helligkeit in %**
- kopiere im Zwischenspeicher Pixel** **nach Pixel**
- kopiere im Zwischenspeicher Pixel** **nach Pixel**
- kopiere im Zwischenspeicher Pixel** **nach Pixel**
- sende Pixel aus Zwischenspeicher an LED-Strip**
- zähle Dimmer von 1 bis** **alle**
- dimme Pixel** **im Zwischenspeicher um %**