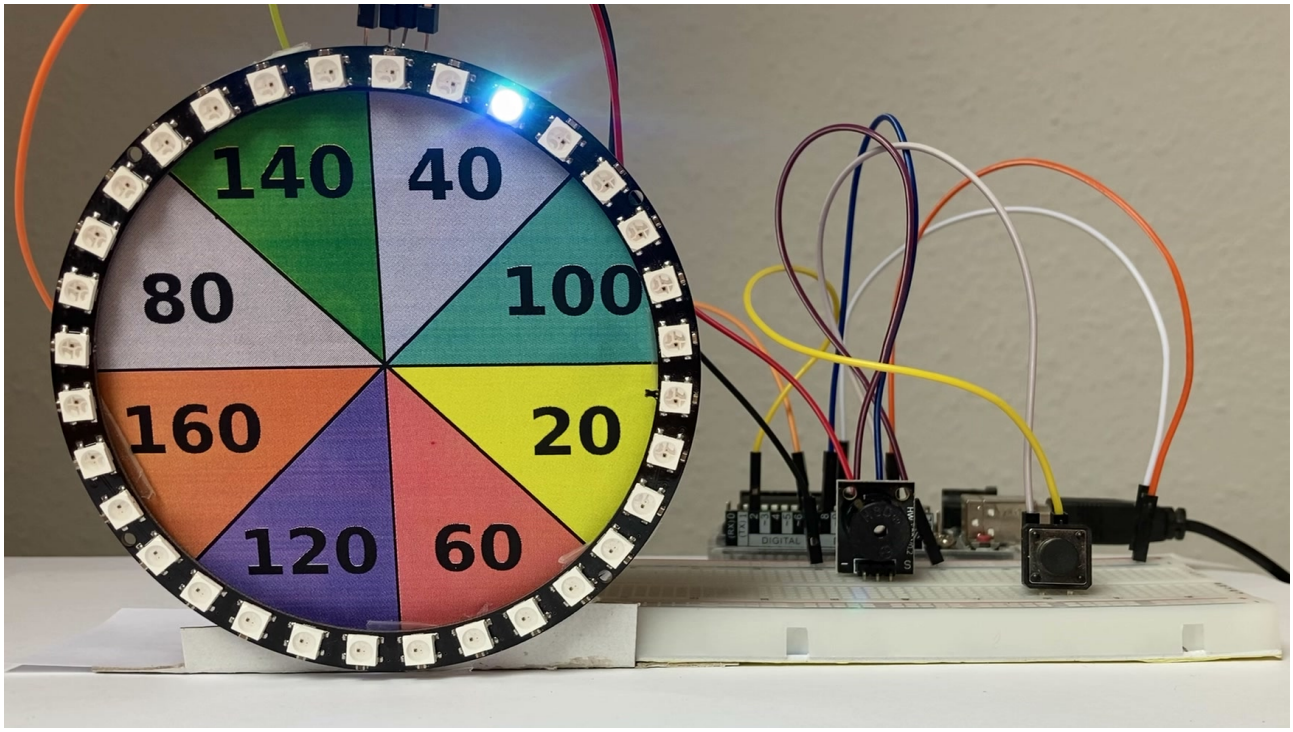
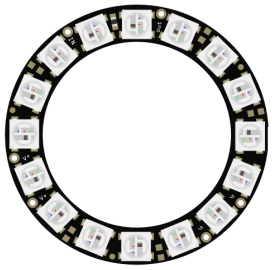
Glücksrad mit einem Neopixel-Ring

Ein Neopixel soll als Glücksrad funktionieren. Wird der Taster gedrückt, läuft das Glücksrad, ein erneuter Tasterdruck stoppt es.





Der NeoPixel-Ring besteht aus mehreren miteinander verbundenen RGB-LEDs. Jede besitzt einen eigenen Controller und kann einzeln angesteuert werden. Er benötigt nur einen digitalen Eingang.

Der NeoPixel-Ring ist in verschiedenen Bauformen zwischen 12 und 60 LEDs erhältlich.

Die Programmierung unterscheidet sich nicht.

RGB ist eine Mischung der Farben Rot, Grün und Blau. Jede Farbe kann von 0 bis 255 gesetzt werden, die Werte werden durch Kommata getrennt.

Beispiele:

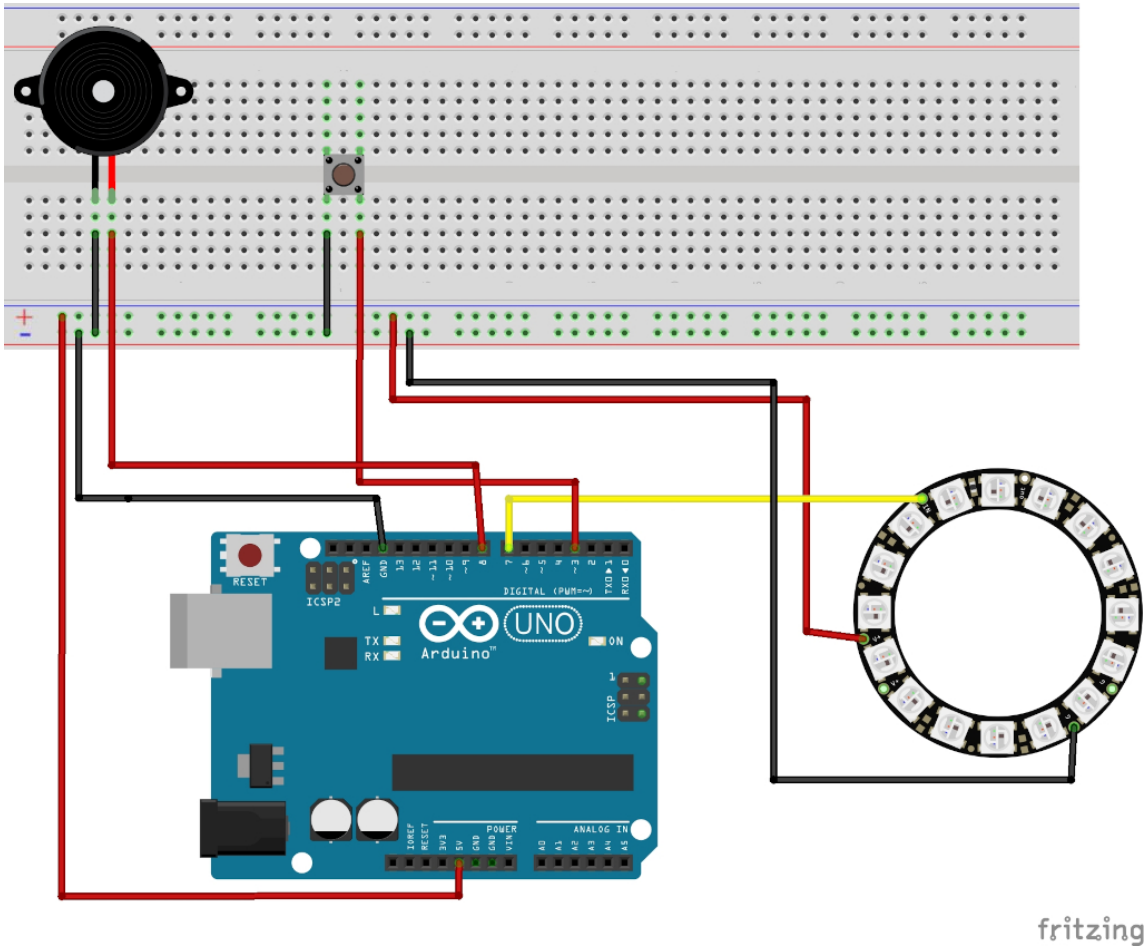
163, 0 ,93	226, 176, 50	255, 255, 0	255, 228, 225

Weitere Informationen

Benötigte Bauteile:

- NeoPixel-Ring
- Taster
- Lautsprecher
- Leitungsdrähte

Baue die Schaltung auf.



Jede LED kann einzeln angesprochen werden.

Die Zählung beginnt mit 0!

Benötigte Bibliothek:

Sketch → Bibliothek einbinden → Bibliotheken verwalten

Übersicht über die Befehle der Bibliothek Adafruit_NeoPixel (Auswahl)

Befehl (Parameter)	**Funktion**
begin()	LED-Ring starten
numPixels()	Anzahl der LEDs lesen
show()	LED-Ring einschalten
clear()	LED-Ring ausschalten
setPixelColor(Nummer, rot, grün, blau)	Farbe einer LED setzen Nummer → Nummer der LED rot -> 0 - 255 grün -> 0 - 255 blau -> 0 - 255
setBrightness()	Helligkeit setzen (0−255)
Color(rot, grün, blau)	**Farbe für alle LEDs setzen** rot -> 0 - 255 grün -> 0 - 255 blau -> 0 – 255 Beispiel blau: int Farbe = LEDRing.Color(0, 0, 255);
fill(Farbe, Start, Ende)	Farbe für die mit Start und Ende bezeichneten Pixel setzen

Probiere die folgenden Beispiele:

Farbwechsel

```
# include <Adafruit_NeoPixel.h>

# define RING 7

// Anzahl der LEDs → muss angepasst werden
# define AnzahlLED 32

// LEDRing -> Name des LED-Rings
Adafruit_NeoPixel LEDRing = Adafruit_NeoPixel(AnzahlLED, RING, NEO_GRB + NEO_KHZ800);

void setup()
{
  // setBrightness(0..255)
  LEDRing.setBrightness(200);

  // NeoPixel Bibliothek initialisieren
  LEDRing.begin();
}

void loop()
{
  // Variable für die Farbe festlegen
  int Farbe;

  // rot
  Farbe = LEDRing.Color(255, 0, 0);
  LEDRing.fill(Farbe, 0, AnzahlLED);
  LEDRing.show();
  delay(1000);

  // grün
  Farbe = LEDRing.Color(0, 255, 0);
  LEDRing.fill(Farbe, 0, AnzahlLED);
  LEDRing.show();
  delay(1000);

  // blau
  Farbe = LEDRing.Color(0, 0, 255);
  LEDRing.fill(Farbe, 0, AnzahlLED);
  LEDRing.show();
  delay(1000);

  // gelb
  Farbe = LEDRing.Color(255, 255, 0);
  LEDRing.fill(Farbe, 0, AnzahlLED);
  LEDRing.show();
  delay(1000);

  // pink
  Farbe = LEDRing.Color(255, 20, 147);
  LEDRing.fill(Farbe, 0, AnzahlLED);
  LEDRing.show();
```

```
  delay(1000);

  // Pause
  LEDRing.clear();
  LEDRing.show();
  delay(2000);
}
```

Lauflicht:

```
# include <Adafruit_NeoPixel.h>

int RING = 7;

// Anzahl der LEDs -> muss angepasst werden
int AnzahlLED = 32;

// LED-Ring -> Name des LED-Rings
Adafruit_NeoPixel LEDRing = Adafruit_NeoPixel(AnzahlLED, RING, NEO_GRB + NEO_KHZ800);

void setup()
{
  // NeoPixel Bibliothek initialisieren
  LEDRing.begin();

  // setBrightness(0..255)
  LEDRing.setBrightness(200);
}

void loop()
{
  // vorwärts mit blau
  for (int LEDNummer = 0; LEDNummer < LEDRing.numPixels(); LEDNummer++)
  {
    LEDRing.setPixelColor(LEDNummer, LEDRing.Color(0, 0, 255));
    LEDRing.show();
    delay(100);
    LEDRing.setPixelColor(LEDNummer, LEDRing.Color(0, 0, 0));
    LEDRing.show();
  }

  // rückwärts mit gelb
  for (int LEDNummer = LEDRing.numPixels(); LEDNummer >= 0 ; LEDNummer--)
  {
    LEDRing.setPixelColor(LEDNummer, LEDRing.Color(255, 255, 0));
    LEDRing.show();
    delay(100);
    LEDRing.setPixelColor(LEDNummer, LEDRing.Color(0, 0, 0));
    LEDRing.show();
  }
}
```

Jeder Taster soll eine Folge von leuchtenden LEDs in verschiedenen Farben auslösen. Dabei soll der jeweils andere Taster den Programmablauf unterbrechen und „seine“ Folge leuchtender LEDs starten.

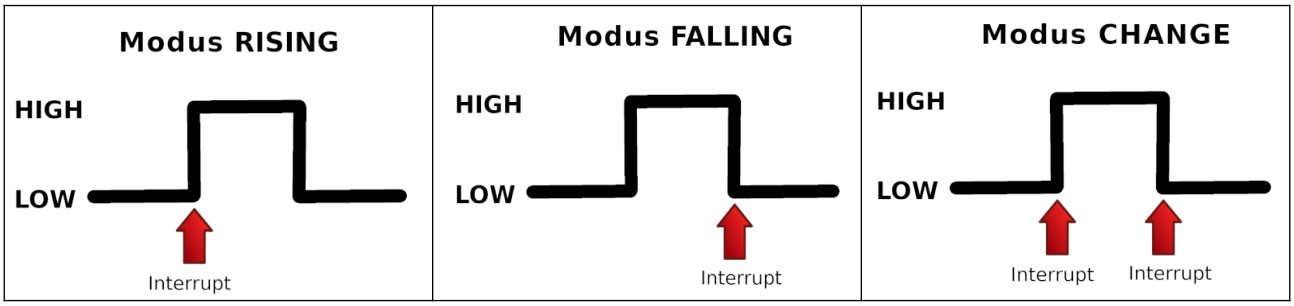
Ein Programmablauf kann nur unterbrochen werden, wenn dem Taster ein Interrupt zugeordnet wird.

Die Taster werden einem Interrupt zugeordnet (`attachInterrupt`). Wenn der Taster betätigt wird, löst er den Interrupt aus. Der normale Programmablauf wird unterbrochen und die festgelegte Methode (Interrupt-Service-Routine) wird ausgeführt. Anschließend wird das Programm normal fortgesetzt.

```
attachInterrupt(digitalPinToInterrupt(TASTER), LEDSchalten, FALLING);
```

Der Taster löst den Interrupt aus, die Interrupt-Service-Routine `LEDSchalten` wird aufgerufen. Der Interrupt soll auf einen Wechsel des Tasterzustands (LOW oder HIGH) reagieren.
Es gibt verschiedene Ereignisse, die den Interrupt auslösen können:

RISING	der Interrupt wird ausgelöst wenn sich der Status von LOW zu HIGH ändert
FALLING	der Interrupt wird ausgelöst wenn sich der Status von HIGH zu LOW ändert
CHANGE	der Interrupt wird ausgelöst wenn sich der Status ändert



Die Taster müssen zwingend am Pin 2 und Pin 3 angeschlossen werden.

Zusätzlich muss die Variable, die eine Statusänderung anzeigt, als `volatile` definiert werden. Variable werden im RAM und temporär im Speicherregister gespeichert, bearbeitet oder verändert.

Es kann vorkommen, dass der Zustand der Variablen im Flashspeicher und im SRAM durch eine neue Wertzuweisung für kurze Zeit nicht übereinstimmen.
Das Schlüsselwort `volatile` weist das Programm an, die Variable immer aus dem Flashspeicher und nicht vom SRAM zu laden. Damit wird garantiert, dass der jeweils aktuelle Wert geladen wird.

Außerdem sind bei der Programmierung einer Interrupt-Methode einige Regeln zu beachten:

- Der Programmteil muss so kurz wie möglich sein.
- Verwende kein `delay()`.
- Benutze auch kein `Serial.print()`.

Das Programm lässt eine Vielzahl von Einstellungen zu:

- die Geschwindigkeit der laufenden LED (Wartezeit)
- der Lautsprecher kann ein- oder aus geschaltet werden (Toene)
- die Farbe der LEDs kann nach jeder Runde oder nach jeder LED neu bestimmt werden (FarbwechselRunde)

Definiere die Variablen und binde die benötigte Bibliothek ein. Beachte die Kommentare.

```
# include <Adafruit_NeoPixel.h>

// digitaler Pin LED-Ring
# define RING 7

// digitaler Pin des Lautsprechers
# define LAUTSPRECHER 8

// Anzahl der LEDs -> muss an den verwendeten Neopixel-Ring angepasst werden
# define AnzahlLED 32

// LED-Ring -> Name des LED-Rings
Adafruit_NeoPixel LEDRing = Adafruit_NeoPixel(AnzahlLED, RING, NEO_GRB + NEO_KHZ800);

// Status entscheidet, ob das Glücksrad läuft oder gestoppt wird
// false -> Glücksrad läuft beim Start
volatile bool Status = false;

// true -> Lautsprecher ein
// false -> Lautsprecher aus
bool Toene = true;

// true -> zufällige Farbe nach jeder Runde wechseln
// false -> Farbwechsel nach jedem Pixel
bool FarbWechselRunde = false;

// Definition der Farben
int ROT, GRUEN, BLAU;

// Taster am Interrupt-Port definieren
# define TASTER 2

// Variablen für die mit millis() ermittelte Zeit
// damit nur jeweils ein Tasterdruck erkannt wird
static unsigned long GesicherteStartZeit = 0;
unsigned long Startzeit;

// Zeit zwischen den Bewegungen der LED
int WarteZeit = 50;
```

Der setup-Teil. Beachte die Kommentare.

```
void setup()
{
  // NeoPixel Bibliothek initialisieren
  LEDRing.begin();

  // setBrightness(0..255)
  LEDRing.setBrightness(250);

  // pinMode des Tasters mit eingeschaltetem Vorwiderstand
  pinMode(TASTER, INPUT_PULLUP);

  // Interrupt dem Taster zuordnen und die Methode LEDSchalten aufrufen
  attachInterrupt(digitalPinToInterrupt(TASTER), LEDSchalten, FALLING);

  // Zufallsgenerator starten
  randomSeed(analogRead(0));
}
```

Der loop-Teil. Beachte die Kommentare.

```
void loop()
{
  // LED-Ring ausschalten
  LEDRing.clear();

  // zufällige Farbwerte Wechsel nach jeder 360° Drehung
  // -> FarbWechselRunde true
  if (FarbWechselRunde)
  {
    ROT = ZufallsFarbe();
    GRUEN = ZufallsFarbe();
    BLAU = ZufallsFarbe();
  }

  // wenn Status false -> Glücksrad läuft
  if (!Status)
  {
    if (Toene) tone(LAUTSPRECHER, 1000, 5);

    // Start der Drehung mit LED 0
    // Ende mit Anzahl der LEDs
    for (int i = 0; i < LEDRing.numPixels(); i ++)
    {
      // zufällige Farbwerte: Wechsel nach jeder LED
      // -> FarbWechselRunde false
      if (!FarbWechselRunde)
      {
        ROT = ZufallsFarbe();
        GRUEN = ZufallsFarbe();
        BLAU = ZufallsFarbe();
      }
      // Farbe für die aktuelle LED
      LEDRing.setPixelColor(i, ROT, GRUEN, BLAU);
```

```
      // LEDs einschalten
      LEDRing.show();

      /*
        Taster gedrückt: Status = true -> Glücksrad stoppen
        break -> for-Schleife verlassen
      */
      if (Status)
      {
        if (Toene) tone(LAUTSPRECHER, 1000, 500);
        break;
      }

      // Wartezeit bis zur nächsten LED
      delay(WarteZeit);

      // aktuell eingeschaltete LEDs ausschalten
      LEDRing.setPixelColor(i, 0);
    }
  }
}
```

Jetzt fehlen noch die Methoden `LEDSchalten()` und `ZufallsFarbe()`.

```
void LEDSchalten()
{
  // aktuelle Zeit sichern
  Startzeit = millis();

  /*
    bei mehr als 250 Millisekunden Differenz
    zwischen der aktuellen und der gesicherten Zeit
    -> Status umdrehen
    aus false wird true -> Glücksrad stoppen
    aus true wird false -> Glücksrad starten
  */
  if (Startzeit - GesicherteStartZeit > 250)
  {
    Status = !Status;
  }

  GesicherteStartZeit = Startzeit;
}
```

```
int ZufallsFarbe()
{
  int Farbe = random(0, 255);
  return Farbe;
}
```

Hartmut Waller Letzte Änderung: 05.10.22