

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

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

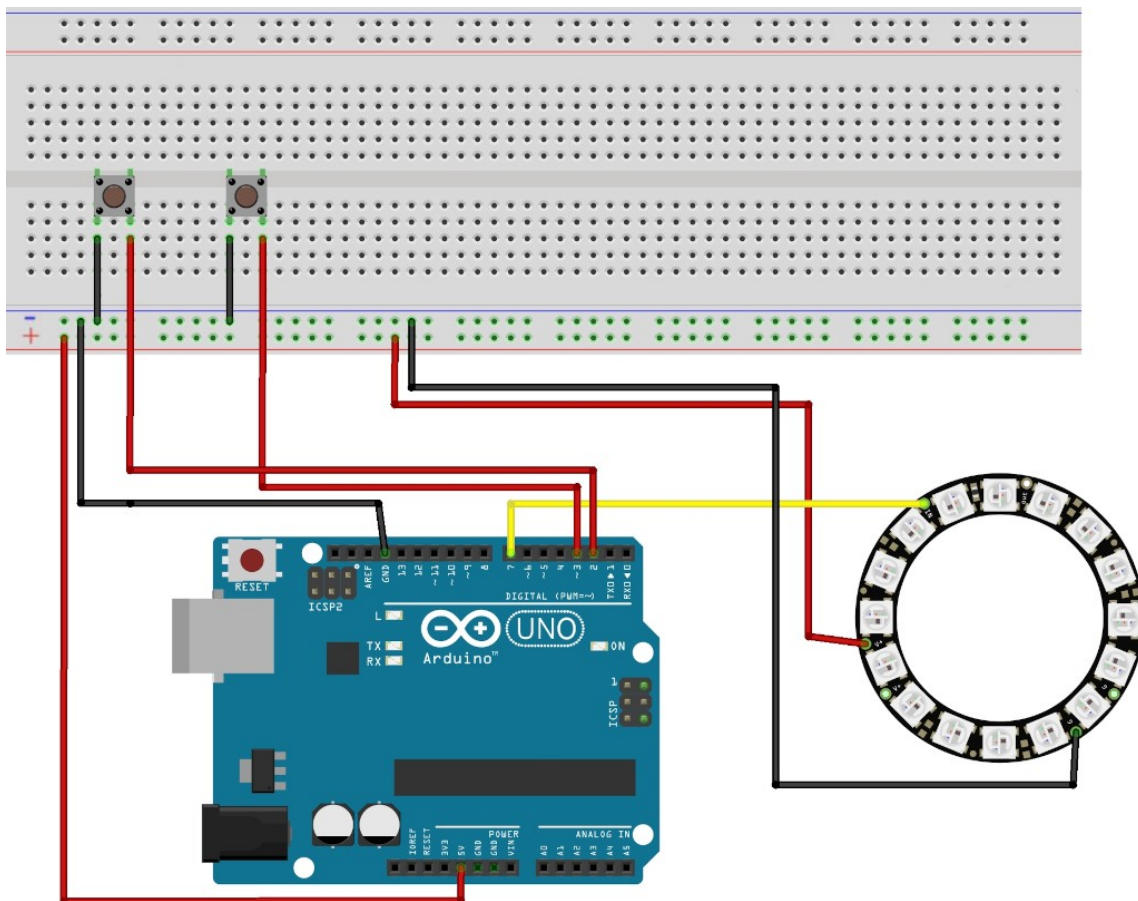


Weitere Informationen

Benötigte Bauteile:

- ➔ NeoPixel-Ring
- ➔ Leitungsdrähte

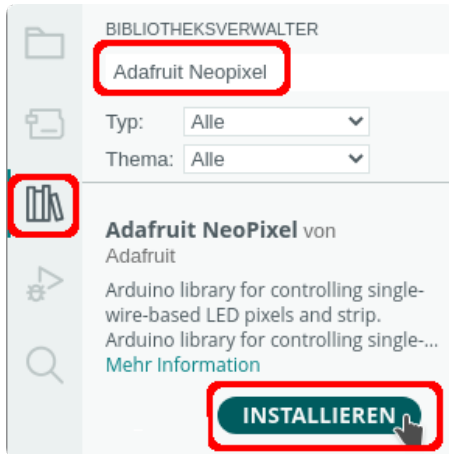
Baue die Schaltung auf.



Jede LED kann einzeln angesprochen werden.

Die Zählung beginnt mit 0!

Benötigte Bibliothek:



Übersicht über die Funktionen der Bibliothek Adafruit_NeoPixel (Auswahl)

Schlüsselwort	Aktion
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 oder: Wert „für gepacktes RGB“ ermitteln long Farbe = LEDRing.Color(0, 0, 255);

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

    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 jedem Taster ein Interrupt zugeordnet wird.



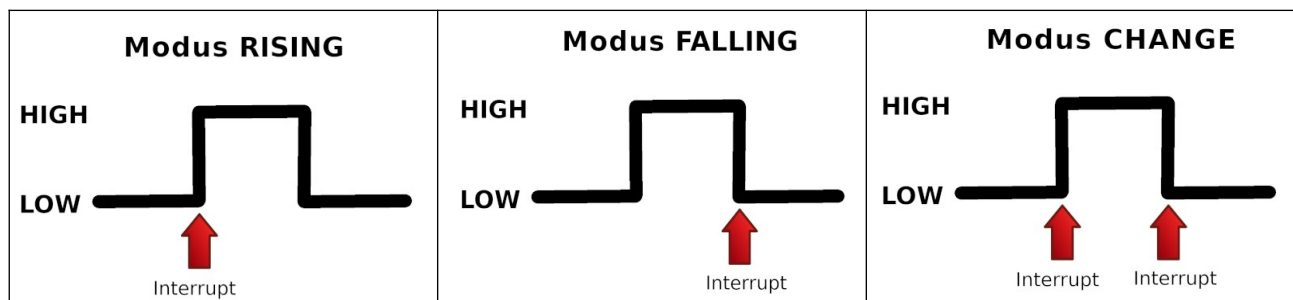
Die Taster werden einem Interrupt zugeordnet (`attachInterrupt`). Wenn einer der Taster betätigt wird, löst er den Interrupt aus. Der normale Programmablauf wird unterbrochen und die festgelegte Methode (Interrupt-Service-Routine) wird ausgeführt. Anschließend wird das Programm normal fortgesetzt.

```
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 eine Variable, die im Hauptprogramm und in Interruptroutinen verwendet wird, als **volatile** definiert werden. Variable werden im SRAM und temporär in internen Registern des Prozessors gespeichert, bearbeitet oder verändert. Es kann vorkommen, dass der Wert der Variablen im SRAM durch eine neue Wertzuweisung für kurze Zeit nicht stimmt, da der letzte Wert zunächst nur in den Prozessorregistern liegt.

Das Schlüsselwort **volatile** weist das Programm an, die Variable immer aktuell im SRAM zu halten. 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()`.

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

```
# include <Adafruit_NeoPixel.h>

# define RING 7

// Anzahl der LEDs -> muss angepasst werden
# define AnzahlLED 32

// Variable für den gedrückten Taster
volatile bool Links = false;
volatile bool Rechts = false;

// Minimum/Maximum für die Farbwerte festlegen
# define Minimum 0
# define Maximum 255

// Leuchtdauer der LEDs
# define Wartezeit 50

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

// Taster definieren
# define TASTERLINKS 2
# define TASTERRECHTS 3
```

Der setup-Teil. Beachte die Kommentare.

```
void setup()
{
    // Zufallsgenerator starten
    randomSeed(analogRead(0));

    // NeoPixel Bibliothek initialisieren
    LEDRing.begin();

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

    LEDRing.clear();
    LEDRing.show();

    pinMode(TASTERLINKS, INPUT_PULLUP);
    pinMode(TASTERRECHTS, INPUT_PULLUP);

    // Methode für das Auslösen des Interrupts definieren
    attachInterrupt(digitalPinToInterrupt(TASTERLINKS), LEDSchaltenLinks, CHANGE);
    attachInterrupt(digitalPinToInterrupt(TASTERRECHTS), LEDSchaltenRechts, CHANGE);
}
```

Der loop-Teil. Beachte die Kommentare.

```
void loop()
{
    // linke Taste gedrückt
    if (Links)
    {
        // zufällige Farbwerte
        int ROT = ZufallsFarbe();
        int GRUEN = ZufallsFarbe();
        int BLAU = ZufallsFarbe();

        /*
        feste Farbwerte:
        ROT:
        int ROT = 255;
        int GRUEN = 0;
        int BLAU = 0;
        -----
        GRUEN
        int ROT = 0;
        int GRUEN = 255;
        int BLAU = 0;
        -----
        BLAU
        int ROT = 0;
        int GRUEN = 0;
        int BLAU = 255;
        -----
        GELB
        int ROT = 255;
        int GRUEN = 255;
        int BLAU = 0;
        */

        for (int LEDNummer = 0; LEDNummer <= LEDRing.numPixels() ; LEDNummer++)
        {
            LEDRing.setPixelColor(LEDNummer, LEDRing.Color(ROT, GRUEN, BLAU));
            LEDRing.show();

            // Unterbrechung wenn der rechte Taster gedrückt wurde
            if (Rechts) break;

            delay(Wartezeit);
        }

        // zurück LEDs ausschalten
        for (int LEDNummer = LEDRing.numPixels(); LEDNummer >= 0 ; LEDNummer--)
        {
            LEDRing.setPixelColor(LEDNummer, 0);
            LEDRing.show();

            // Unterbrechung wenn der rechte Taster gedrückt wurde
            if (Rechts) break;
            delay(Wartezeit);
        }
    }
}
```

```

    Links = false;
    LEDRing.clear();
    LEDRing.show();
}

// rechte Taste gedrückt
if (Rechts)
{
    int ROT = ZufallsFarbe();
    int BLAU = ZufallsFarbe();
    int GRUEN = ZufallsFarbe();

    for (int LEDNummer = LEDRing.numPixels(); LEDNummer >= 0 ; LEDNummer--)
    {
        LEDRing.setPixelColor(LEDNummer, LEDRing.Color(ROT, BLAU, GRUEN));
        LEDRing.show();

        // Unterbrechung wenn der rechte Taster gedrückt wurde
        if (Links) break;
        delay(Wartezeit);
    }

    // LEDs ausschalten
    for (int LEDNummer = LEDRing.numPixels(); LEDNummer >= 0 ; LEDNummer--)
    {
        LEDRing.setPixelColor(LEDNummer, 0);
        LEDRing.show();

        // Unterbrechung wenn der linke Taster gedrückt wurde
        if (Links) break;
        delay(Wartezeit);
    }

    Rechts = false;
    LEDRing.clear();
    LEDRing.show();
}
}

```

Jetzt fehlen noch die Methoden LEDSchaltenRechts() und LEDSchaltenLinks(). Sie tun nichts anderes als den Zustand der Taster zu markieren.

```

void LEDSchaltenRechts()
{
    Rechts = true;
    Links = false;
}

void LEDSchaltenLinks()
{
    Links = true;
    Rechts = false;
}

```