Lauflicht mit attachInterrupt schalten

Bei dieser Aufgabe lernst du ein neues Konzept für die Definition von Variablen vom Typ int kennen.

enum kann die Definition einer größeren Anzahl Variablen vom Typ int vereinfachen. Das erste Element erhält den Wert 0, jedes weitere wird um 1 hochgezählt.

Das Beispielprogramm definiert die Wochentage als Aufzählung.

```
enum Wochentage
{
  Sonntag,
  Montag,
  Dienstag,
  Mittwoch,
  Donnerstag,
  Freitag,
  Samstag
};

void setup()
{
  Serial.begin(9600);
  // Startwert → Sonntag, Endwert → Samstag
  for (int i = Sonntag; i <= Samstag; i++)
  {
    Serial.print(i); Serial.print("\t");
  }
  Serial.println();
  Serial.print("So\tMo\tDi\tMi\tDo\tFr\tSa");
}

void loop()
{
  // bleibt leer, Programm läuft nur einmal
}
```

Der Startwert des ersten Elements kann auch gesetzt werden:

```
enum Farben
{
  GRUEN = 3,
  WEISS,
  ROT,
  BLAU,
  GELB
};

void setup()
{
  Serial.begin(9600);
  // Startwert → GRUEN, Endwert → GELB
  for (int i = GRUEN; i <= GELB; i++)
  {
    Serial.print(i);
    Serial.print("\t");
  }
}

void loop()
{
   // bleibt leer, Programm läuft nur einmal
}
```

Ein weiteres Beispiel: Ein Lauflicht:

```
enum Farben
{
  GRUEN = 3,
  WEISS,
  ROT,
  BLAU,
  GELB
};

void setup()
{
  for (int i = GRUEN; i < GELB; i++)
  {
    pinMode(i, OUTPUT);
  }
}
```

```
void loop()
{
  for (int i = GRUEN; i <= GELB; i ++)
  {
    // aktuelle LED i einschalten
    digitalWrite(i, HIGH);
    delay(200);

    // aktuelle LED i ausschalten
    digitalWrite(i, LOW);
  }
}
```

Außerdem benötigst du eine Erweiterung der for-Schleife.
for-Schleifen können mit dem Schlüsselwort break beendet werden:

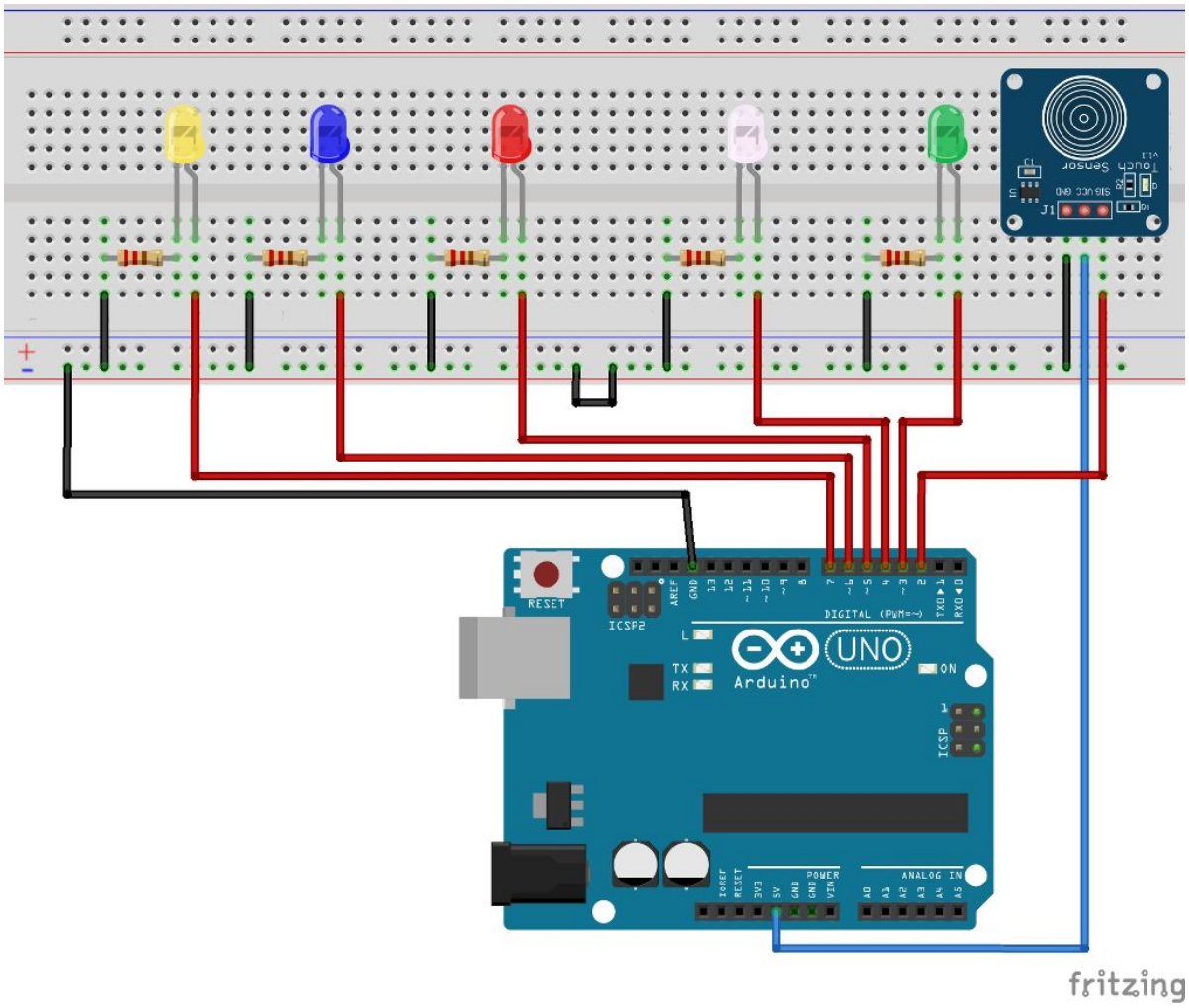
Beispiel: die for-Schleife wird nach dem Wert 5 beendet.

```
void setup()
{
  Serial.begin(9600);
  for (int i = 0; i < 10; i++)
  {
    Serial.println("Zahl " + String(i));
    if (i == 5)
    {
      Serial.println("Schleife wird nach " + String(i) + " beendet!");
      break;
    }
  }
}

void loop()
{
  // bleibt leer, Programm läuft nur einmal
}
```

Benötigte Bauteile:

- 5 LEDs
- 5 Widerstände > 100 Ω
- Berührungssensor
- Leitungsdrähte

Baue die Schaltung auf:



Beim Start des Programms sollen die LEDs nacheinander für 200 Millisekunden nacheinander leuchten. Eine Berührung des Sensors schaltet das Lauflicht aus, die nächste Berührung startet es von Neuem.

Definiere die LEDs als Aufzählung, der Sensor muss zwingend an Pin 2 oder 3 angeschlossen werden. Die Variable, deren Zustand für das Stopp und das Weiter des Lauflichtlichts verantwortlich ist, wird als bool definiert.

```
enum Farben
{
  GRUEN = 3,
  WEISS,
  ROT,
  BLAU,
  GELB
};

int SENSOR = 2;

volatile bool Stopp = true;
```

Im setup-Teil wird der pinMode der LEDs definiert. Der Startwert ist die grüne LED, der Endwert die gelbe LED.
Außerdem wird die Methode LEDSchalten als Interrupt-Routine definiert. Weil der beim Berühren des Sensors ausgelesenen Wert HIGH ist, reagiert der Interrupt auf RISING.

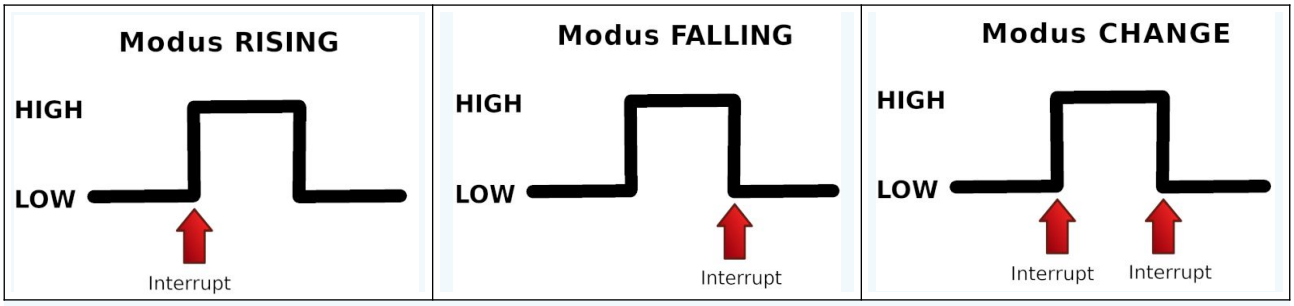
```
void setup()
{
  // pinMode der LEDs definieren
  for (int i = GRUEN; i < GELB; i++)
  {
    pinMode(i, OUTPUT);
  }

  // Methode für das Auslösen des Interrupts definieren
  attachInterrupt(digitalPinToInterrupt(SENSOR), LEDSchalten, RISING);
}
```

Der Taster wird dem 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



Der Taster muss zwingend am Pin 2 oder 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()`.

Der loop-Teil. Beachte die Kommentare.

```
void loop()
{
  // solange Stopp den Wert true hat
  while (Stopp)
  {
    for (int i = GRUEN; i <= GELB; i ++)
    {
      // aktuelle LED i einschalten
      digitalWrite(i, HIGH);
      delay(200);

      // aktuelle LED i ausschalten
      digitalWrite(i, LOW);

      // Schleifen-Durchlauf stoppen (break)
      // wenn die Variable Stopp den Wert false hat
      if (!Stopp) break;
    }
  }
}
```

Jetzt benötigst du nur noch die Methode LEDSchalten.
Sie tut nichts anderes als den Wert für Stopp jeweils umzudrehen.

```
void LEDSchalten()
{
  Stopp = !Stopp;
}
```