Übungen mit LEDs

Vor Beginn der Arbeiten solltest du die ***Elektrotechnischen Grundlagen*** aus folgenden Dokumenten durchlesen und verstanden haben:

- **Wie funktioniert ein Steckbrett**
- **Diode, LED**

Als ***Vorwiderstand*** einer LED am Arduino kannst du jeden Widerstandswert von ***etwa 250Ω bis etwa 1kΩ*** benutzen.

Achte auf die richtige Polung der LED, sonst leuchtet sie nicht :-)

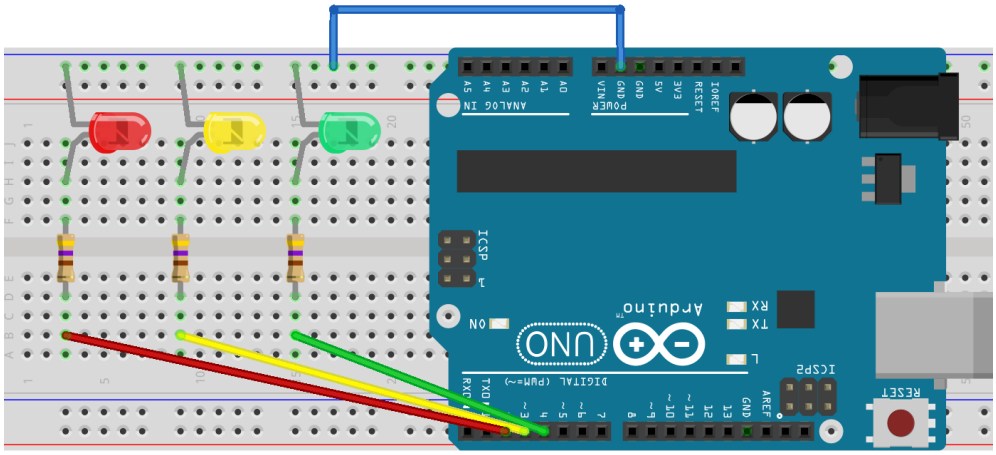
Projekt Ampel

Hardware

Du benötigst

1 Steckbrett
3 LEDs *(rot, gelb, grün)*
3 Vorwiderstände *(Wert: s. Oben)*
4 Kabel *(male/male)*
1 Arduino (Uno)

Aufbau

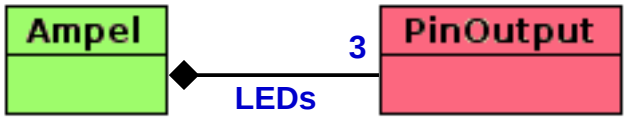


Stecke die kurzen Beine der LEDs am Steckbrett oben auf die blaue Leiste und ziehe von dort ein Kabel zu einem ***GND-Pin*** (Minus-Pol der Stromversorgung).
Stecke die langen Beine der LEDs wie im Bild ins Steckbrett.

Verbinde ein Ende der Widerstände wie im Bild mit dem langen Bein der LEDs. Vom anderen Ende der Widerstände führst du je ein Kabel zu den ***Pins 2 (rot), 3 (gelb) und 4 (grün)***.

Grundlegende Anforderungen an das Programm

Als ***Vorlagen*** benötigst du ***die Klassen Arduino*** und ***PinOutput***.



1. Erstelle eine ***Klasse Ampel***, die drei Referenz-Attribute vom Typ ***PinOutput*** hat und nenne sie ***'rot'***, ***'gelb'*** und ***'gruen'***.
2. Erstelle im ***Konstruktor*** die zugehörigen Objekte an den Pins 2 (rot), 3 (gelb) und 4 (gruen). Schalte alle LEDs im Konstruktor ein, damit du siehst, ob du alles richtig verkabelt hast.
   Tipp: *Sieh dir vorher die Klassenkarte der Klasse PinOutput an und nutze die Code-Vervollständigung (Strg + Leertaste).*
3. Schreibe nun die ***Methoden*** `allesAn()`, `allesAus()`, `rot()`, `rot_gelb()`, `gelb()`, `gruen()` und fülle ihren Rumpf mit entsprechendem Code.
4. Erzeuge ein Objekt der Klasse Ampel und teste deine Methoden.
   (Starte vor dem Testen immer in BlueJ die Virtuelle Maschine oder BlueJ selbst neu !!!)

Bringe Ordnung in die Ampel-Phasen

Bisher kannst du die Ampel-Phasen schalten, wie du willst. In der Wirklichkeit schaltet eine 3-Lichter-Ampel in der Reihenfolge: rot → rot_gelb → gruen → gelb … und wieder von vorne ...

5. Schreibe eine weitere ***Methode*** `ampel_zyklus()`, die der Reihe nach folgendes macht:

ROT	warte 5 Sekunden
ROT_GELB	warte 1 Sekunde
GRUEN	warte 3 Sekunden
GELB	warte 1 Sekunde
ROT	

   Tipp: *Die Klasse Arduino bietet dir eine statische Methode* `pause(int milliSekunden)`*. Statische Methoden kann man aufrufen, ohne dass man ein Objekt dieser Klasse referenzieren muss. Man benutzt dann bei der Punktnotation einfach den Klassennamen statt eines Objektnamens:* `Arduino.pause(...)`
6. Erzeuge ein Objekt der Klasse Ampel und teste die neue Methode.
7. Schreibe eine letzte ***Methode*** `ampel_zyklen(int anzahl)`, die einen ganzzahligen Übergabe-Parameter entgegen nimmt. In ihrem Rumpf soll mit Hilfe einer gezählten Wiederholung die Methode `ampel_zyklus()` entsprechend oft aufgerufen werden.
   Denkfrage: *Was macht diese Methode, wenn du ihr einen negativen Wert übergibst? Erst nachdenken, dann ausprobieren, ob du Recht hast … :-)*
8. *Erzeuge ein Objekt der Klasse Ampel und teste die neue Methode.*

Dimmen einer LED

Eigentlich stellt man die Helligkeit einer LED über ihren Vorwiderstand ein. Diesen können wir aber nicht durch unser Programm ändern.

Es gibt aber einen Trick:
Man schaltet die LED so schnell hintereinander an und wieder aus, dass unser Auge das nicht mehr wahrnehmen kann. Das Auge nimmt stattdessen eine verminderte Helligkeit wahr. Die wahrgenommene Helligkeit kann man dann einstellen, indem man die An-Phase im Verhältnis zur Aus-Phase unterschiedlich lang macht.

Hierzu bietet das Arduino spezielle Pins, die sogenannten ***PWM-Pins (Pulsweiten-Modulation)***.
Man kann in Prozent angeben, wie lange der HIGH-Anteil sein soll: 0 bedeutet ganz aus, 100 bedeutet ganz an, die Werte dazwischen dimmen entsprechend …
Als PWM-Pins stehen beim Arduino Uno nur die Pins ***3, 5, 6, 9, 10 und 11*** zur Verfügung.

9. Starte zunächst in BlueJ die virtuelle Maschine neu oder schließe BlueJ und öffne es erneut. Erzeuge nun durch Rechts-Klick ein Objekt der ***Klasse PinPWM*** an Pin 3 (gelbe LED der Ampel) und experimentiere interaktiv mit den Methoden dieser Klasse und beobachte das Verhalten deiner gelben LED.

10. Starte in BlueJ erneut die virtuelle Maschine neu oder schließe BlueJ und öffne es wieder. Erstelle eine neue ***Klasse DimmLED***, die von PinPWM erbt.
    Rufe im Konstruktor den Super-Konstruktor mit mit Pin 3 auf.
    Schreibe dann die beiden ***Methoden*** `an()` und `aus()`, die den DutyCycle auf 100% bzw. 0% setzen.

11. Erzeuge ein Objekt der Klasse DimmLED und teste deine Methoden.

12. Schreibe die ***Methoden*** `heller()` und `dunkler()`, die die Helligkeit jeweils um 20% vergrößern bzw. verringern.
    Verständnis-Frage: *Was geschieht, wenn dadurch der Helligkeits-Wert über 100% oder unter 0% gesetzt wird?*
    (Sieh dir zum Verständnis den Code der Klasse PinPWM an.)

13. Erzeuge ein Objekt der Klasse DimmLED und teste deine Methoden.

14. Schreibe die ***Methoden*** `aufdimmen()` bzw. `abdimmen()`, die langsam die Helligkeit von 0% auf 100% bzw. umgekehrt mit Hilfe einer gezählten Wiederholung verändern.
    Tipp: *Die Klasse Arduino bietet dir eine statische Methode* `pause(int milliSekunden)`.
    Statische Methoden kann man aufrufen, ohne dass man ein Objekt dieser Klasse referenzieren muss. Man benutzt dann bei der Punktnotation einfach den Klassennamen statt eines Objektnamens: `Arduino.pause(...)`

15. *Erzeuge ein Objekt der Klasse DimmLED und teste deine Methoden.*