UE 14 Einführung in den ARDUINO

Klicke oben auf das Video-Vorschaubild und informiere Dich über den ARDUINO!

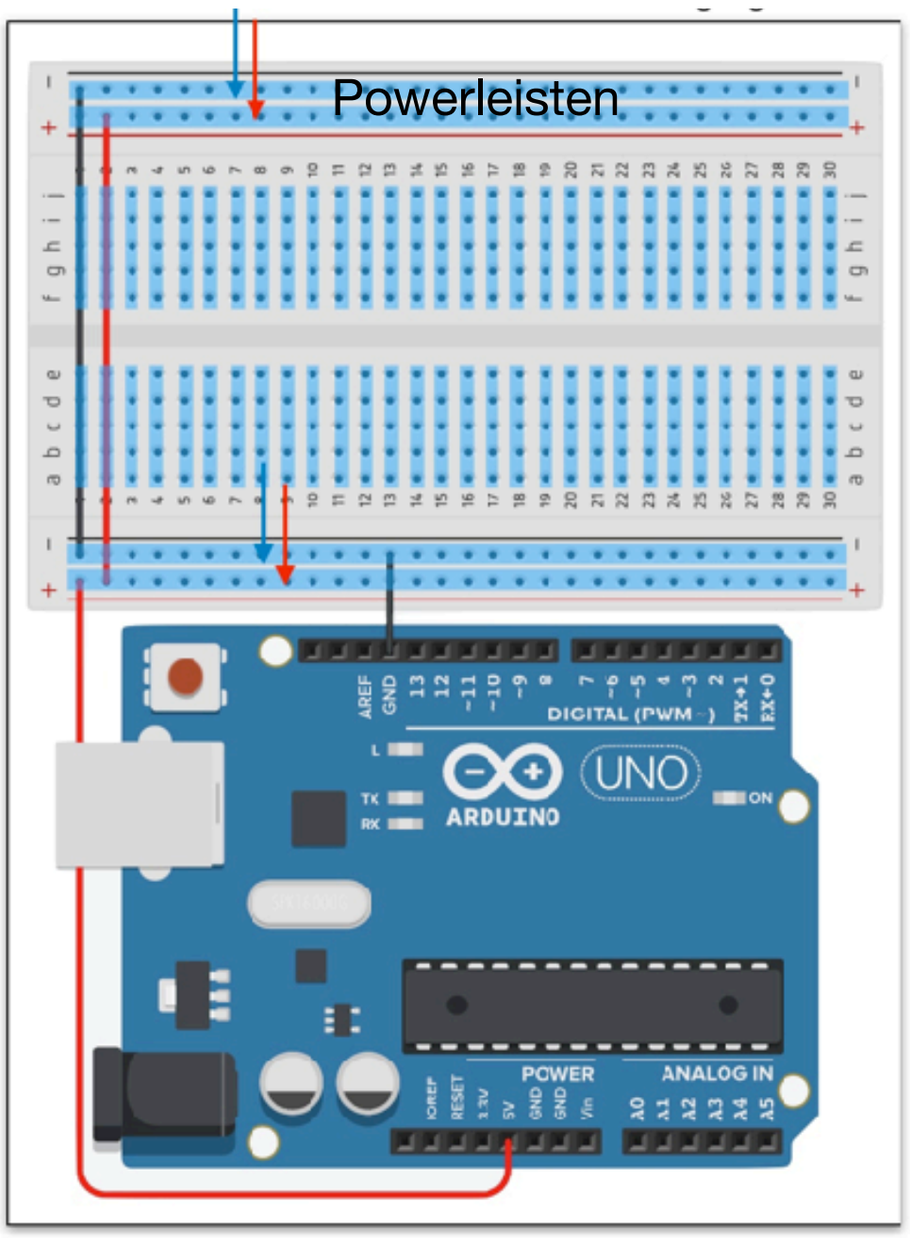
Inhalt:

Um eine Ampel an einer gleichberechtigten Kreuzung mit einer roten, grünen und gelben LED nachzubilden, können Sie den folgenden Beispielcode für einen Arduino verwenden:

Um ein Arduino-Programm zu schreiben, das einen Servomotor von der Position 0° langsam auf 90° innerhalb von 10 Sekunden bewegt, dann 10 Sekunden wartet und wieder auf 0° zurückfährt und nach weiteren 45 Sekunden die Bewegung wiederholt, können Sie die `Servo` -Bibliothek verwenden, die es Ihnen ermöglicht, einen Servomotor einfach zu steuern. Hier ist ein Beispielcode:

ARDUINO und Breadboard

Für die Versuche mit dem ARDUINO benötigt man einen ARDUINO mit USB-Kabel und einen Computer -sowie eine Experimentierplatte (auch Breadboard genannt).



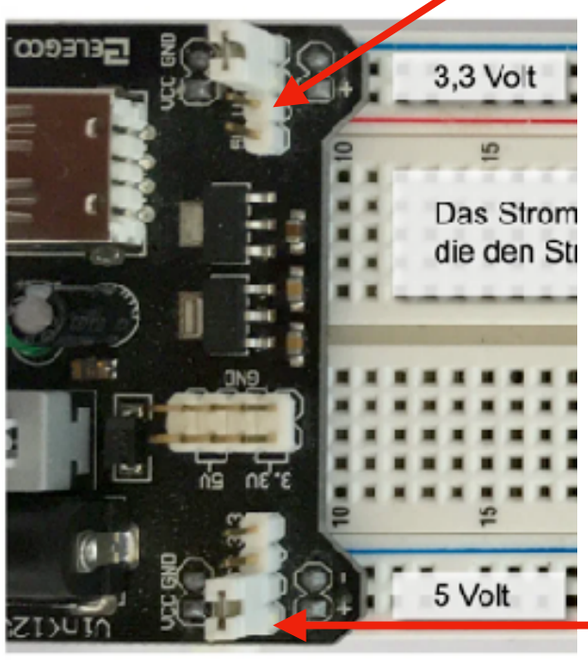
Das Breadboard ist eine Steckplatine mit vielen Steckkontakten im Abstand von 2,54 mm, in die man elektrische Bauteile stecken und miteinander verbinden kann ohne zu löten. Es gibt sie in unterschiedlichen Größen.

Hier wird ein mittelgroßes Breadboard über den ARDUINO mit Gleichspannung 5 V versorgt.

Die blaue Markierungen zeigen, wie die Kontakte miteinander verbunden sind.

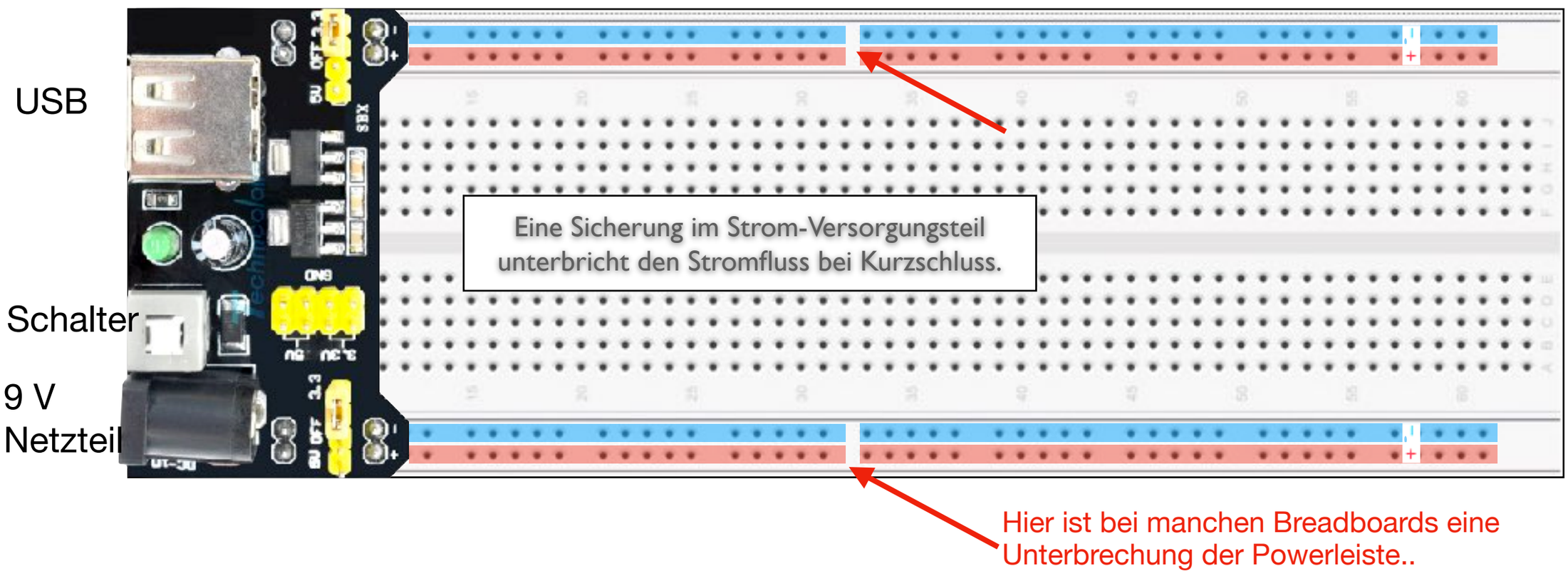
Im ARDUINO ist eine Sicherung integriert, die bei Kurzschluss den Stromfluss unterbindet.

Jumper zur Einstellung der Spannung



Alternativ zum ARDUINO geschieht die Stromversorgung mit dem aufsteckbaren Strom-Versorgungsteil. Welche Spannung (5 V oder 3,3 V) ausgegeben wird, lässt sich unabhängig für die Spannungsausgänge einstellen, indem man den Jumper auf die zugeordnete Spannung steckt. Strom liefert ein USB-Kabel oder 9V-Netzteil.

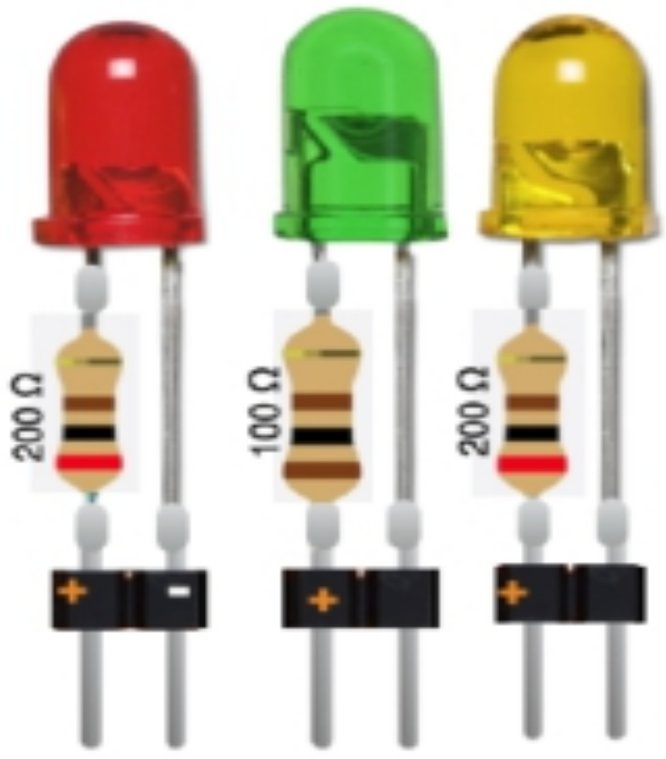
Jumper zur Einstellung der Spannung



Leuchtdiode (LED) in der 5 mm-Bauform

Eine Leuchtdiode (kurz LED für: Light Emitting Diode) ist ein elektronisches Halbleiter-Bauelement. Fließt durch die Diode Strom in Durchlass-Richtung, so strahlt sie Licht, Infrarot-Strahlung oder auch Ultraviolett-Strahlung ab.

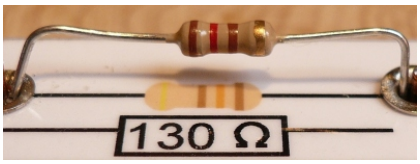
Anders als Glühlampen erzeugen Leuchtdioden Licht in einer bestimmten Farbe und geben weniger Wärme ab. (Sie benötigen weniger Energie.)
Eine LED darf höchstens mit 1,5 Volt Spannung betrieben werden, sonst “brennt sie durch”.
Deshalb wird in den Versuchen immer ein Widerstand in den Stromkreis geschaltet.



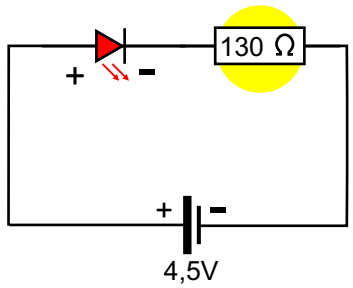
Wenn man verhindern möchte, dass die Leuchtdioden versehentlich oder mutwillig durch zu hohen Stromfluss zerstört werden, kann man diese wie oben präparieren.

Der +Pol der LED liegt auf der Seite mit dem Widerstand.

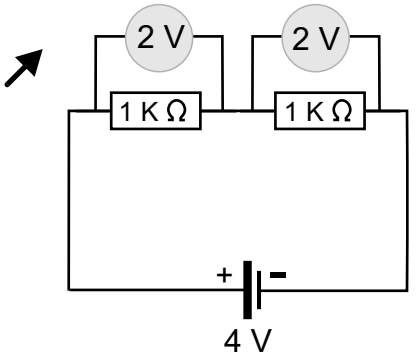
Der Widerstand



Der Widerstand ist eines der wichtigsten Bauelemente in elektronischen Schaltungen. Widerstände werden immer dort eingesetzt, wo eine bestimmte Stromstärke nicht überschritten werden soll (zur Strombegrenzung) oder um aus einer vorliegenden Spannung eine bestimmte Spannung abzuteilen.



Widerstand als Strombegrenzer



Widerstand als Stromteiler

Widerstände können aus den verschiedensten Materialien hergestellt werden, die schlechter leiten als Kupfer, z.B. verschiedene Metallegierungen, Kohlenstoff.
Sie leiten den Strom in beiden Richtungen gleichmäßig gut, ihre Polung ist daher beim Einbau egal. Der Widerstandswert wird in Ohm gemessen und auf den Widerständen in einem Farbcode angegeben.
Neben den Festwiderständen gibt es noch regelbare Widerstände:
Trimmwiderstände oder auch Potentiometer .

Hier im Bausatz werden Drehwiderstände verwendet, deren Widerstandswert durch Drehen mit einem kleinen Schraubendreher verändert werden kann.

Benutzt man einen Außenkontakt und den Mittelkontakt, ist er wie ein normaler Widerstand geschaltet .

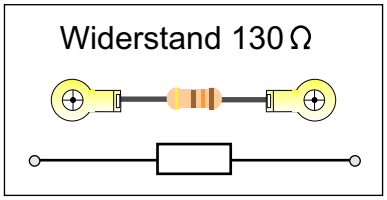
Ist der Trimmwiderstand wie nebenstehend angeschlossen, wirkt er als Spannungsteiler.

Heißleiter (NTC Widerstand) und **Kaltleiter** (PTC -Widerstand) sind temperaturabhängige Widerstände.

Lichtabhängige Widerstände (LDR = Light Dependent Resistor) , auch Fotowiderstand genannt, verändern ihren Widerstandswert je nach Lichteinfall auf die lichtempfindliche Schicht.

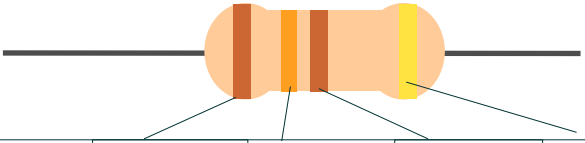
Je mehr Licht auf den LDR fällt, um so geringer ist sein Widerstand.

Der Widerstandsfarbcode

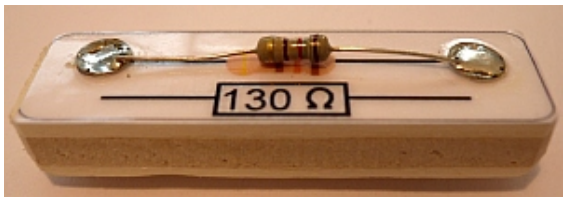


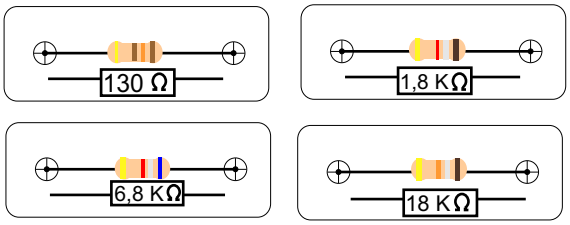
Der 4. Ring (Toleranz) gibt an, wie genau der angegebene Widerstandswert erreicht wird. Die hier verwendeten Widerstände haben zumeist einen goldfarbenen 4. Ring.

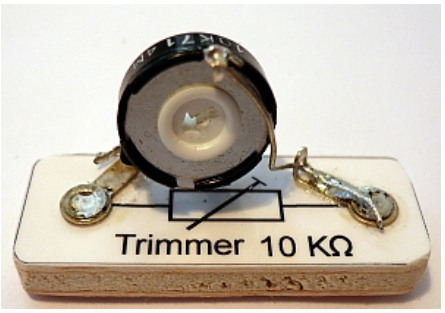
Der Widerstands-Farbcode

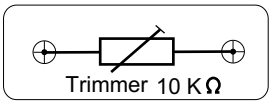


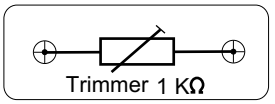
Farbe	1. Ring (1. Ziffer)	2. Ring (2. Ziffer)	3. Ring Multiplikator	4. Ring Toleranz
keine ✕	—	—	—	±20 %
silber	—	—	10^{-2} = 0,01	±10 %
gold	—	—	10^{-1} = 0,1	±5 %
schwarz	—	0	10^{0} = 1	—
braun	1	1	10^{1} = 10	±1 %
rot	2	2	10^{2} = 100	±2 %
orange	3	3	10^{3} =1000	—
gelb	4	4	10^{4}	—
grün	5	5	10^{5}	±0,5 %
blau	6	6	10^{6}	±0,25 %
violett	7	7	10^{7}	±0,1 %
grau	8	8	10^{8}	—
weiß	9	9	10^{9}	—



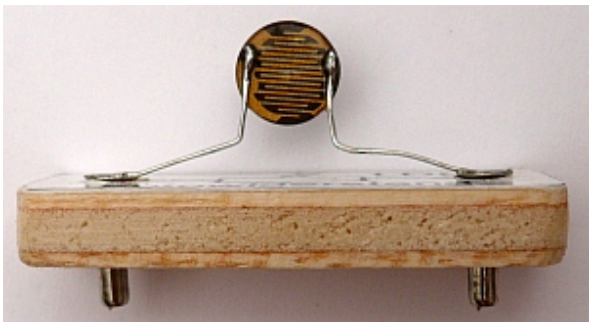


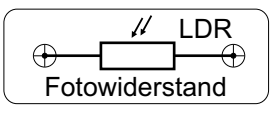


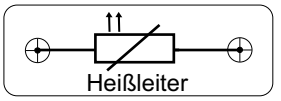




Mittelkontakt und 1 Außenkontakt anschließen!



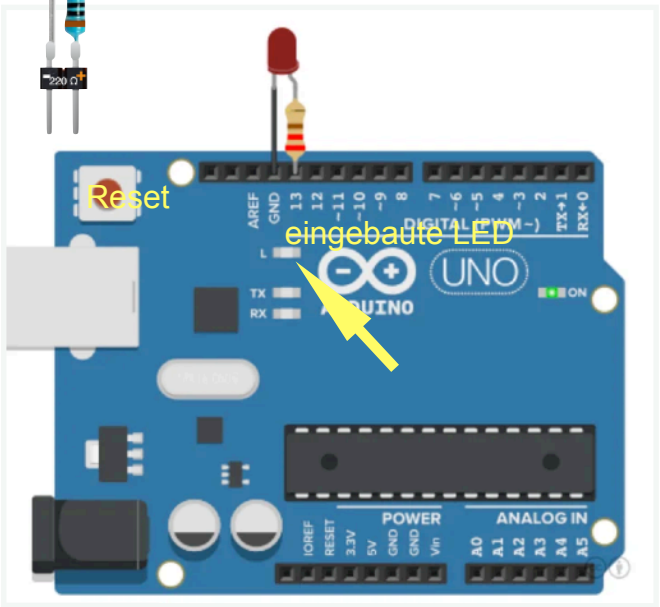




Der Heißleiter ist ein Widerstand, der bei steigender Temperatur kleiner wird.
Bei der Montage zunächst 2 Steckschuhe anlöten und abknicken. Heißleiterkontakte in Steckschuh festklemmen.

mein erstes Programm : LED an PIN 13 programmieren

Setze die LED mit der +Seite auf Pin 13.



Sobald der ARDUINO am Computer angeschlossen ist, beginnt die LED im Takt zu leuchten, genau wie die eingebaute LED.
Wenn das nicht der Fall ist, drücke die Reset-Taste.

Die rote LED soll an Pin 13 blinken.
Dazu schließt man die LED an Pin 13 des Arduino an und teilt dem Arduino in der Setup-Funktion durch den Befehl pinMode(13, OUTPUT) mit, dass er auf diesem Pin die Steuerung für die LED ausgeben soll.
In der Loop-Funktion soll der Arduino dann die LED immer wieder einschalten und ausschalten. Eingeschaltet wird die LED mit digitalWrite(13, HIGH) und ausgeschaltet mit digitalWrite(13, LOW). Mit HIGH wird am Pin 13 die Spannung von 5V bereitgestellt und mit LOW wird die Spannung am Pin 13 wieder auf 0V zurückgesetzt.
Mit dem Befehl delay(1000) wird die Dauer eingestellt, bis die LED umgeschaltet wird.

Achtung:
Jeder Befehl muss mit ; beendet werden!

```
void setup ()
{
pinMode (13, OUTPUT);
 }

void loop ()
{
digitalWrite (13, HIGH);
delay (1000);
digitalWrite (13, LOW);
delay (1000);
}
```

Anweisung, dass Pin 13 als Ausgabe-Pin genutzt wird

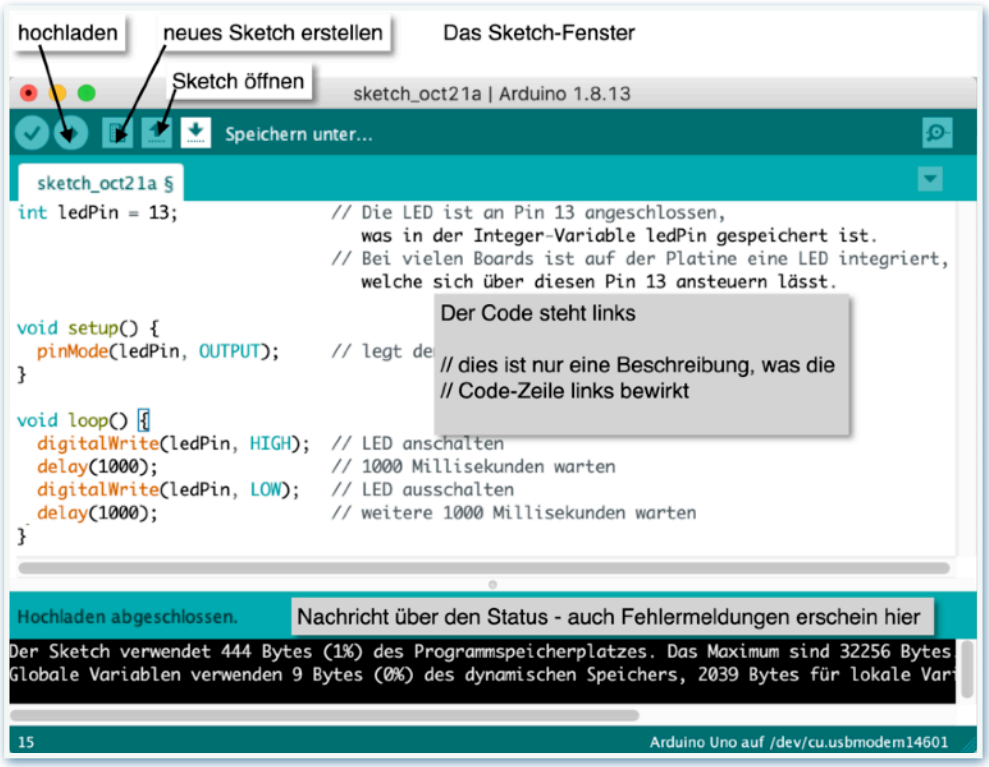
SETUP

Pin 13 auf HIGH setzen = Spannung 5V
Dauer 1000 Millisekunden = 1 sec
Pin 13 auf LOW setzen = Spannung oV
Dauer 1000 Millisekunden = 1 sec

LOOP

wird ständig überholt

Programmieren mit dem Sketch-Fenster



Sketch-Fenster mit Erläuterungen zum Programmcode.

- Starte das Sketch-Fenster und klicke auf „neu“:

- Kopiere den Code und füge ihn in das Programmfenster ein!

```
 void setup ()
{
pinMode (13, OUTPUT);
 }

void loop ()
{
digitalWrite (13, HIGH);
delay (1000);
digitalWrite (13, LOW);
delay (1000);
}
```

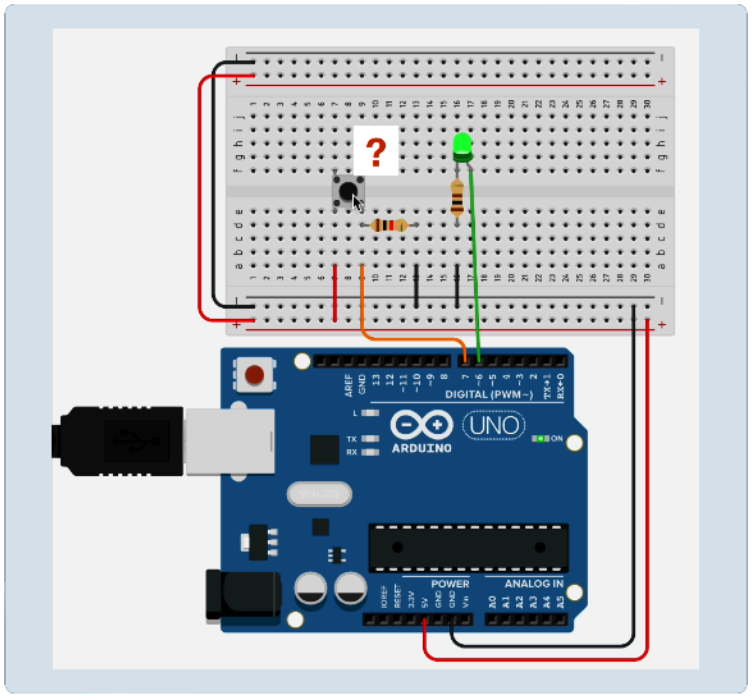
Sketch-Fenster mit dem eingefügten Code

- Lade das Programm hoch

- Verändere die Leuchtdauer im Programm und beobachte die Veränderungen!

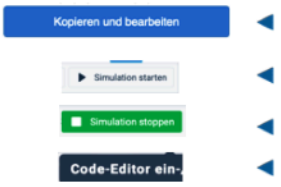
https://www.werken-technik.de/Arduino/arduino-programmieren-mit-taster-und-led.html

LED mit einem Taster einschalten

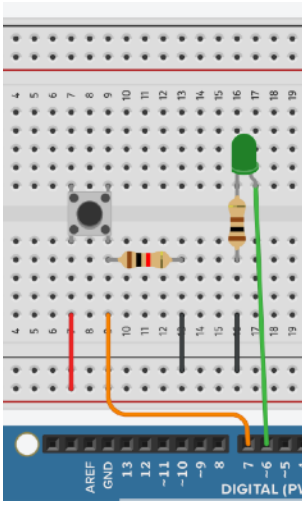


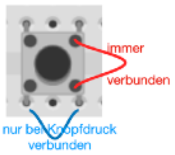
Klicke auf das TINKERCAD-ICON und melde Dich bei TINKERCAD an.

Mit Mausklick gelangst Du dann auf die links abgebildete Schaltung.



Baue die Schaltung auf. Verwende am Schalter einen Widerstand von 1 KOhm.





Der Tastschalter

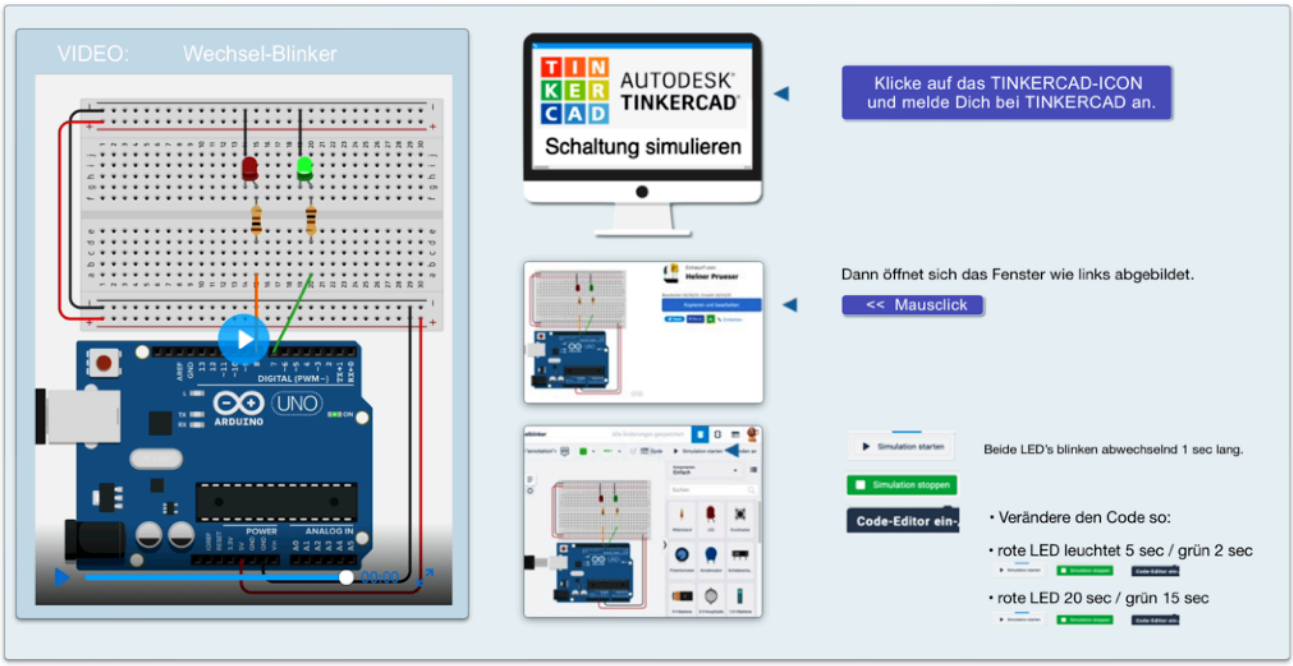
Programm-Code:

```
int LED=6;
int taster=7;
int tasterstatus=0;


  void setup()
{
  pinMode(LED, OUTPUT);
  pinMode(taster, INPUT);
}

void loop()
{
  tasterstatus=digitalRead(taster);
  if (tasterstatus == HIGH)

{
  digitalWrite(LED, HIGH);
  delay(5000); // 5 Sekunden warten

  digitalWrite(LED, LOW);
}
  else
{
  digitalWrite(LED, LOW);

}

}
```

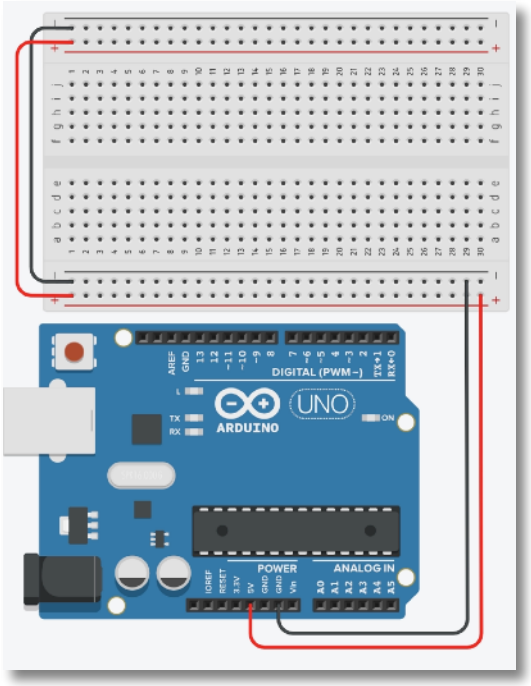
Welche Aufgabe hat der 1 KOhm-Widerstand beim Taster?

Sobald man den Taster drückt, liegt an Pin 7 des ARDUINO eine Spannung an. Lässt man den Taster los, kann es vorkommen, dass Restladungen herumschwirren und der ARDUINO reagiert, als wenn der Taster gedrückt wird.
Um das zu vermeiden, verbindet man den Schalter über einen Widerstand mit 1 Kilo-Ohm mit GND.
Dadurch liegt beim geöffnetem Taster eine negative Spannung an.

2 LED`s blinken im Wechsel



Baue die Schaltung auf : rote LED auf Pin 8 / grüne LED auf Pin 7



Programm-Code: „sketch“:

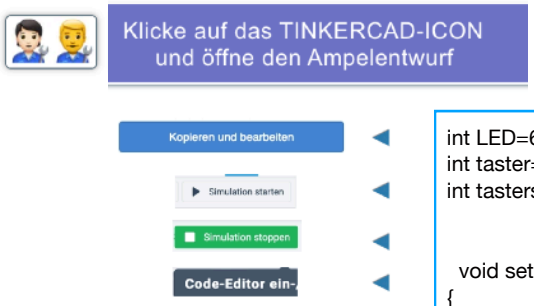
```
 void setup()
{
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
}

void loop()
{
  digitalWrite(7, HIGH);
  delay(1000); // 1 Sekunde Pause
  digitalWrite(7, LOW);

  digitalWrite(8, HIGH);
  delay(1000); // 1 Sekunde Pause
  digitalWrite(8, LOW);

}
```

- Verbinde den ARDUINO mit dem Computer und gib den neuen Sketch ein!
- Lade ihn hoch.
- Machen die LED`s, was beabsichtigt ist?
- Verändere die Blinkzeiten im Programmcode
- lade den veränderten Code hoch

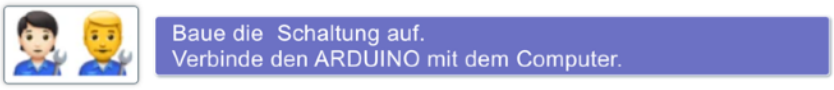
- Verändere den Programm-Code so, dass die grüne LED 2 Sekunden leuchtet und die rote LED 8 sec.
- Mache davon ein Video und sende es an Deine Lehrkraft!

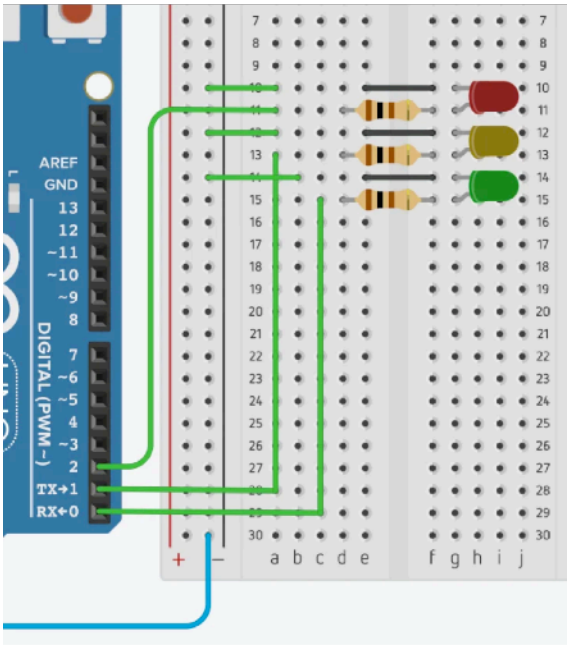
www.werken-technik.de/Arduino/ue20-ampelsteuerung-einfache-ampel.html

Programmierung einer einfachen Ampel



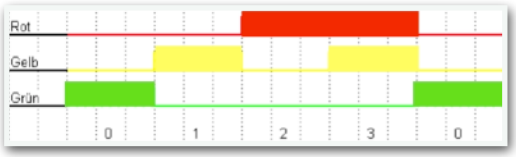
führe diese Schritte aus.
Verändere die Ampelzeiten!





- *Starte das ARDUINO Programmfenster und klicke auf neues Programmfenster:*

- *kopiere oben den Programm-Code*
- *Lade ihn hoch auf den ARDUINO.*
- *Verändere die Blinkzeiten im Programmcode lade den veränderten Code hoch*



Ampelphasen:

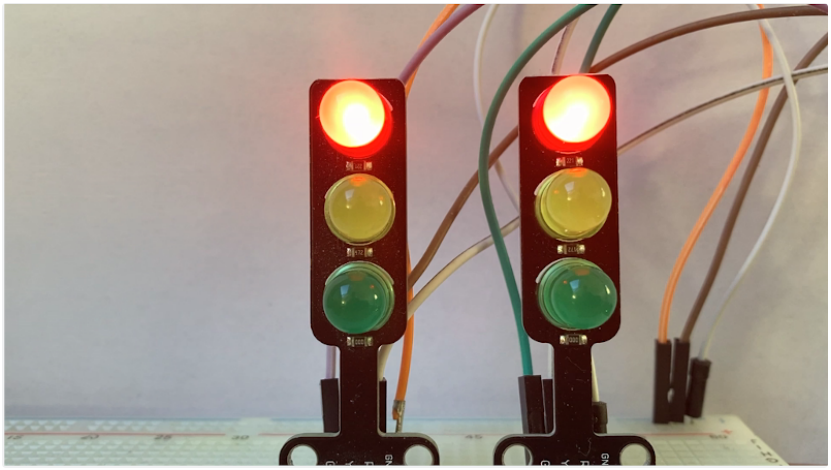
rot	30 Sekunden
rotgelb	4 Sekunde
grün	30 Sekunden
gelb	3 Sekunden

Programm-Code:

```
int LED=6;
int taster=7;
int tasterstatus=0;


 void setup()
{
 pinMode(LED, OUTPUT);
 pinMode(taster, INPUT);
}

void loop()
{
 tasterstatus=digitalRead(taster);
 if (tasterstatus == HIGH)

{
 digitalWrite(LED, HIGH);
 delay(5000); // 5 Sekunden warten

 digitalWrite(LED, LOW);
}
 else
{
 digitalWrite(LED, LOW);

}

}int led_red = 0; // LED rot
int led_yellow = 1; //LED gelb
int led_green = 2; // LED grün

void setup() {
 // set up all the LEDs as OUTPUT
 pinMode(led_red, OUTPUT);
 pinMode(led_yellow, OUTPUT);
 pinMode(led_green, OUTPUT);
}

void loop() {
 // turn the green LED on and the other LEDs off
 digitalWrite(led_red, LOW);
 digitalWrite(led_yellow, LOW);
 digitalWrite(led_green, HIGH);
 delay(4000);    // wait 6 seconds

 // turn the yellow LED on and the other LEDs off
 digitalWrite(led_red, LOW);
 digitalWrite(led_yellow, HIGH);
 digitalWrite(led_green, LOW);
 delay(500);   // wait 1 second

 // turn the red LED on and the other LEDs off
 digitalWrite(led_red, HIGH);
 digitalWrite(led_yellow, LOW);
 digitalWrite(led_green, LOW);
 delay(5000);  // wait 3 seconds

 // turn the yellow LED on and the other LEDs off
 digitalWrite(led_red, LOW);
 digitalWrite(led_yellow, HIGH);
 digitalWrite(led_green, LOW);
 delay(500);   // wait 1 second
}
```

Zwei Ampeln

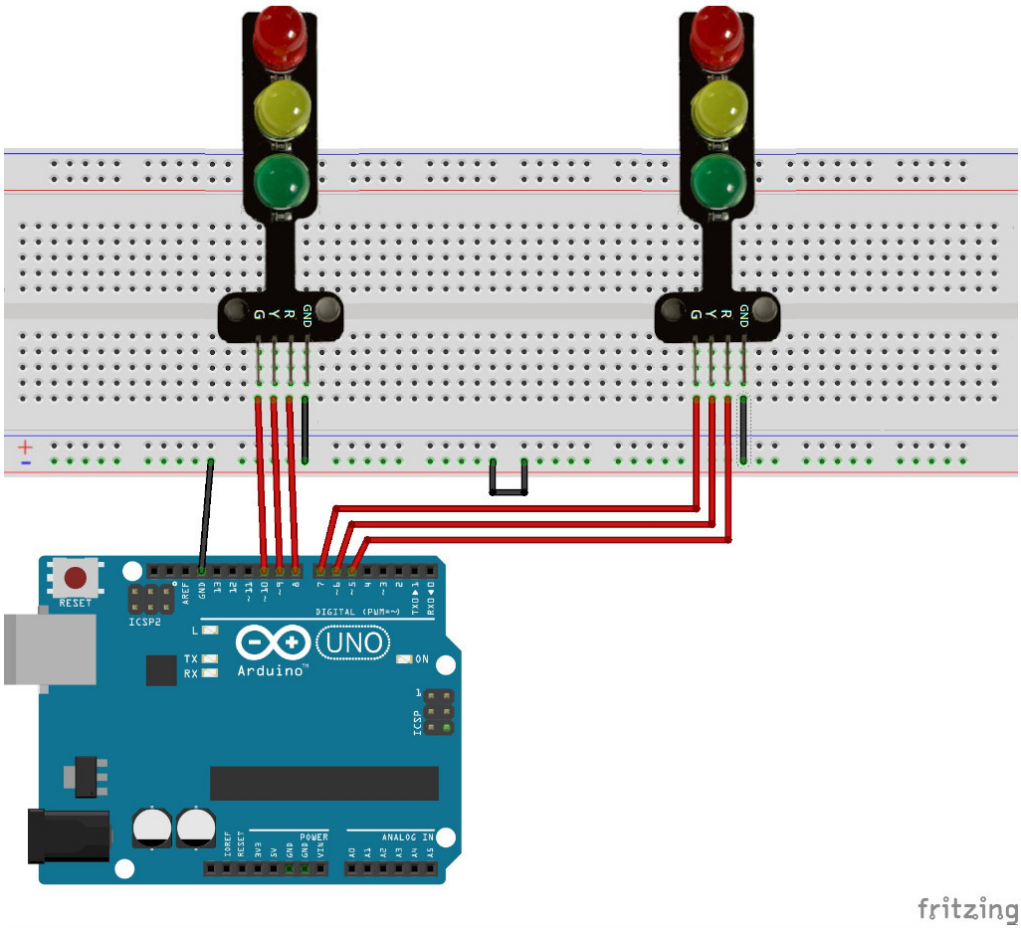
Zwei Ampeln sollen in einem festgelegten Takt geschaltet werden.



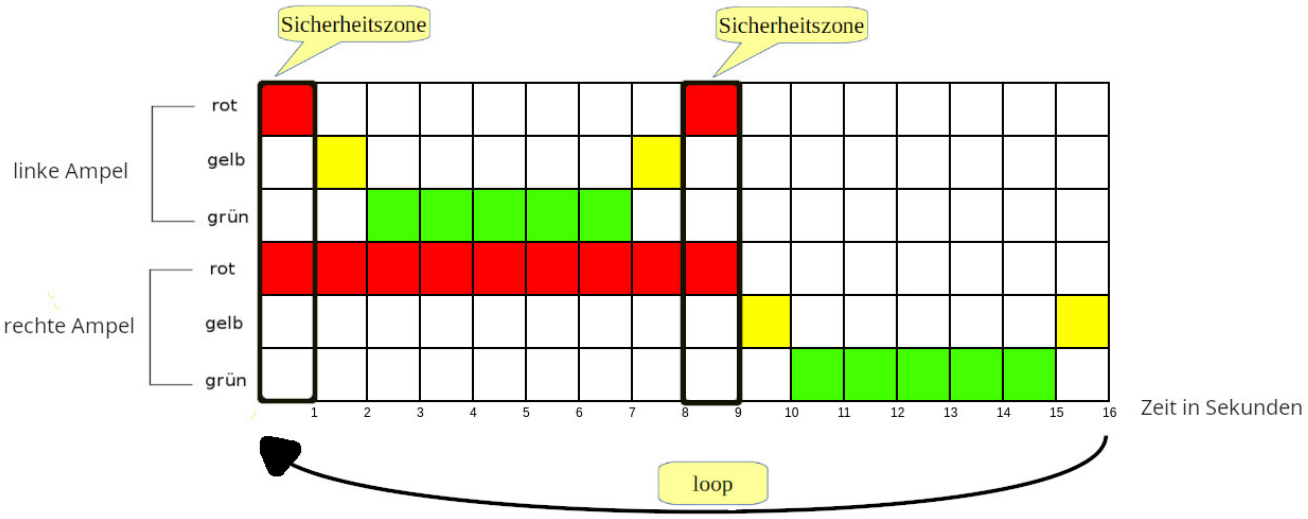
Benötigte Bauteile:

- 2 Ampeln
- Leitungsdrähte

Baue die Schaltung auf:



Der zeitliche Ablauf:



Die Variablen: Sie sorgen dafür, dass im Programmablauf die Zuordnung der LEDs vereinfacht wird.

```
// linke Ampel
int ROT_L = 8;
int GELB_L = 9;
int GRUEN_L = 10;

// rechte Ampel
int ROT_R = 5;
int GELB_R = 6;
int GRUEN_R = 7;
```

Der setup-Teil:

```
void setup()
{
  pinMode(ROT_L, OUTPUT);
  pinMode(GELB_L, OUTPUT);
  pinMode(GRUEN_L, OUTPUT);
  pinMode(ROT_R, OUTPUT);
  pinMode(GELB_R, OUTPUT);
  pinMode(GRUEN_R, OUTPUT);
}
```

Beginne den loop-Teil mit der Sicherheitszone (beide Ampeln zeigen für 1 Sekunde rot).

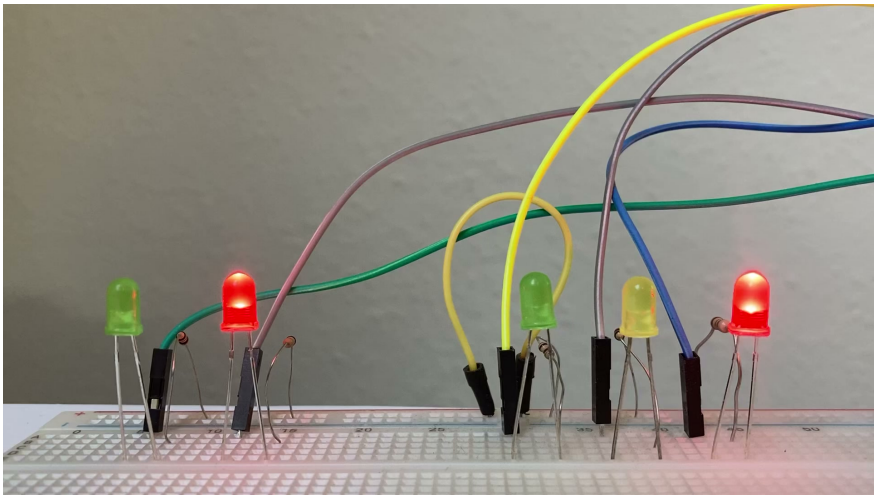
```
void loop()
{
  // beim Start zeigen beide Ampeln rot
  digitalWrite(ROT_L, HIGH);
  digitalWrite(ROT_R, HIGH);
```

```
  // 1 Sekunde Sicherheitszone
  delay(1000);

  // linke Ampel wird über rot/gelb auf grün geschaltet
  // GELB_L -> an, 1 Sekunde warten, GELB_L -> aus, ROT_L -> aus, GRUEN_L -> an
  digitalWrite(GELB_L, HIGH);
  delay(1000);
  digitalWrite(GELB_L, LOW);
  digitalWrite(ROT_L, LOW);
  digitalWrite(GRUEN_L, HIGH);
  delay(5000);

  // linke Ampel wird von grün -> gelb auf rot geschaltet
  // GRUEN_L -> aus, GELB_L -> an, 1 Sekunde warten, GELB_L aus, ROT_L -> an
  digitalWrite(GRUEN_L, LOW);
  digitalWrite(GELB_L, HIGH);
  delay(1000);
  digitalWrite(GELB_L, LOW);
  digitalWrite(ROT_L, HIGH);

  // 1 Sekunde Sicherheitszone, beide Ampeln bleiben rot
  delay(1000);
```

```
  // rechte Ampel über rot-gelb auf grün schalten
  // GELB_R -> an, 1 Sekunde warten, GELB_R -> aus, ROT_R -> aus, GRUEN_R -> an
  digitalWrite(GELB_R, HIGH);
  delay(1000);
  digitalWrite(GELB_R, LOW);
  digitalWrite(ROT_R, LOW);
  digitalWrite(GRUEN_R, HIGH);
  delay(5000);

  // rechte Ampel wird von grün -> gelb auf rot geschaltet
  // GRUEN_R -> aus, GELB_R -> an, 1 Sekunde warten, GELB_R aus,
  // ROT_R wird beim Start von loop ausgeschaltet
  digitalWrite(GRUEN_R, LOW);
  digitalWrite(GELB_R, HIGH);
  delay(1000);
  digitalWrite(GELB_R, LOW);
}
```

Ampel mit Fußgängerampel

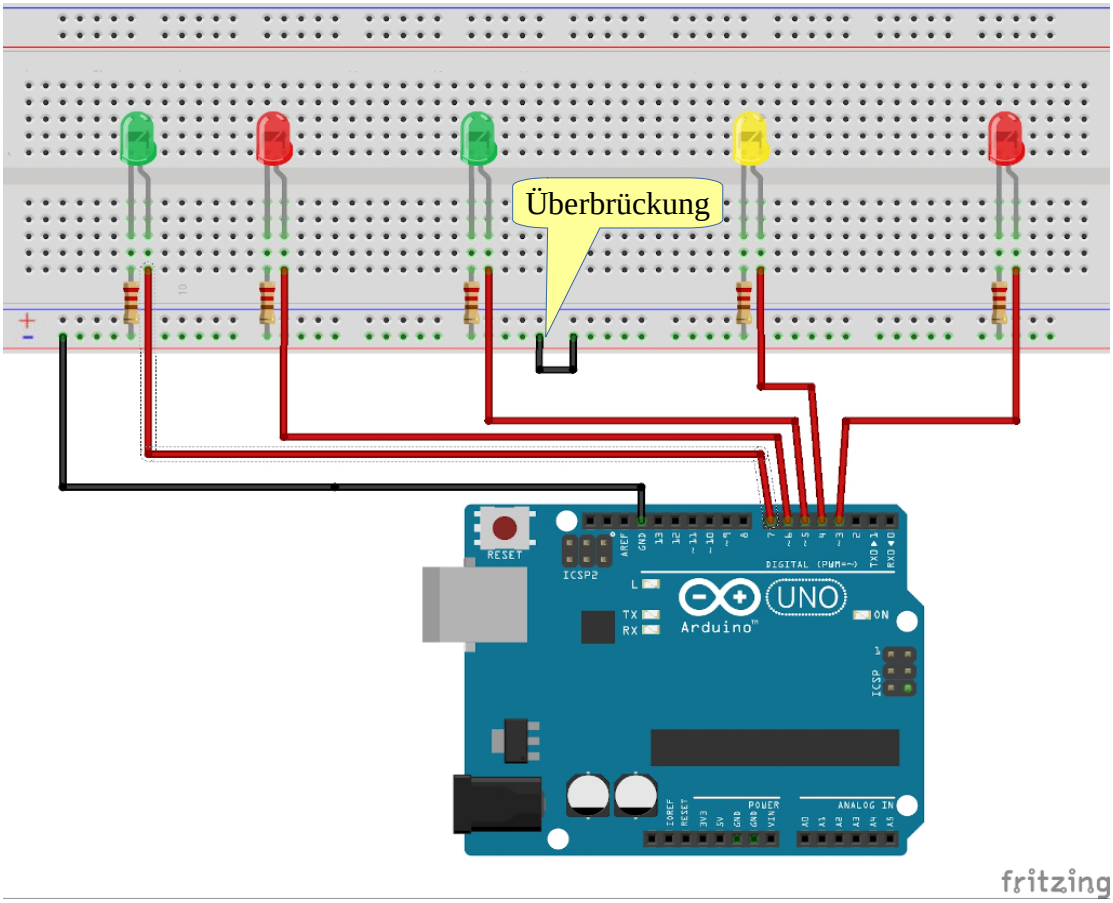
Eine Ampel soll zusammen mit einer Fußgängerampel in einem fesgelegten Takt geschaltet werden.



Benötigte Bauteile:

- 2 rote LEDs
- 2 grüne LEDs
- 1 gelbe LED
- 5 Widerstände > 100 Ω
- Leitungsdrähte

Baue die Schaltung auf:



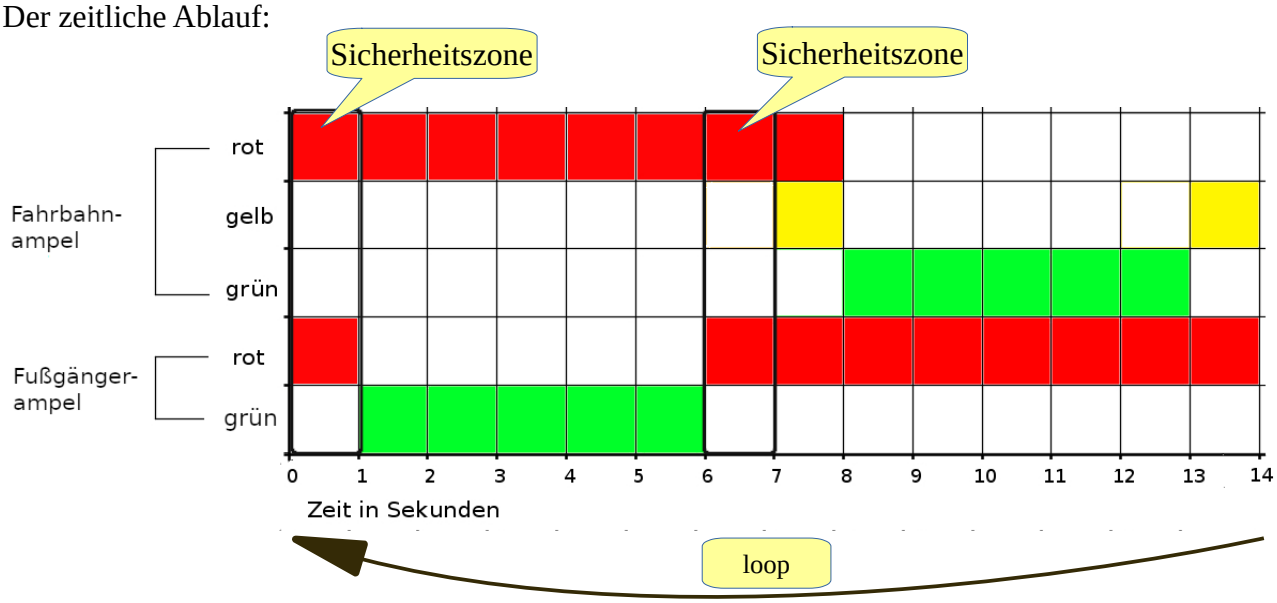
Die Schaltung der Fahrbahnampel:

1 Sekunde Sicherheitszone
5 Sekunden rot
1 Sekunde Sicherheitszone (beide Ampeln zeigen für 1 Sekunde rot)
1 Sekunde rot/gelb
5 Sekunden grün
1 Sekunde gelb

Die Schaltung der Fußgängerampel:

1 Sekunde Sicherheitszone (beide Ampeln zeigen für 1 Sekunde rot)
5 Sekunden grün
1 Sekunde Sicherheitszone (beide Ampeln zeigen für 1 Sekunde rot)
7 Sekunden rot

Der zeitliche Ablauf:



Die Variablen: Sie sorgen dafür, dass im Programmablauf die Zuordnung der LEDs vereinfacht wird.

```
// Fahrbahnampel
int ROT = 3;
int GELB = 4;
int GRUEN = 5;

// Fußgängerampel
int F_ROT = 6;
int F_GRUEN = 7;
```

Füge dem setup-Teil den pinMode für die Fußgängerampel (F_ROT, F_GRUEN) hinzu.

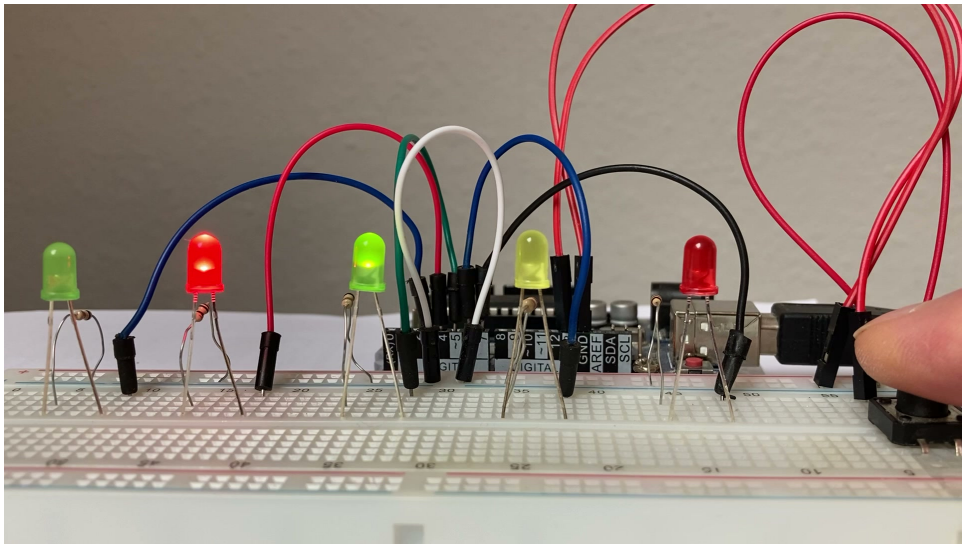
```
void setup()
{
  pinMode(ROT, OUTPUT);
  pinMode(GELB, OUTPUT);
  pinMode(GRUEN, OUTPUT);
  pinMode(F_ROT, OUTPUT);
  pinMode(F_GRUEN, OUTPUT);
}
```

Beginne den loop-Teil mit der Sicherheitszone (beide Ampeln zeigen für 1 Sekunde rot).

```
void loop()
{
  digitalWrite(ROT, HIGH);
  digitalWrite(F_ROT, HIGH);

  // 1 Sekunde Sicherheitszone
  delay(1000);

  // Fußgängerampel wird von rot auf grün geschaltet
  digitalWrite(F_ROT, LOW);
  digitalWrite(F_GRUEN, HIGH);
  delay(5000);

  // Fußgängerampel wird von grün auf rot geschaltet
  // F_GRUEN -&gt; aus, F_ROT -&gt; an
  digitalWrite(F_GRUEN, LOW);
  digitalWrite(F_ROT, HIGH);

  // 1 Sekunde Sicherheitszone, beide Ampeln bleiben rot
  delay(1000);

  // Fahrbahnampel auf rot-gelb und dann auf grün schalten
  // GELB -&gt; an, 1 Sekunde warten, GELB und ROT -&gt; aus, GRUEN -&gt; an
  digitalWrite(GELB, HIGH);
  delay(1000);
  digitalWrite(ROT, LOW);
  digitalWrite(GELB, LOW);
  digitalWrite(GRUEN, HIGH);

  // GRUEN -&gt; 5 Sekunden, GRUEN -&gt; aus, GELB 1 Sekunde an,
  // GELB aus
  delay(5000);
  digitalWrite(GRUEN, LOW);
  digitalWrite(GELB, HIGH);
  delay(1000);
  digitalWrite(GELB, LOW);
}
```

Ampel mit Fußgängerampel und Tasterbedienung

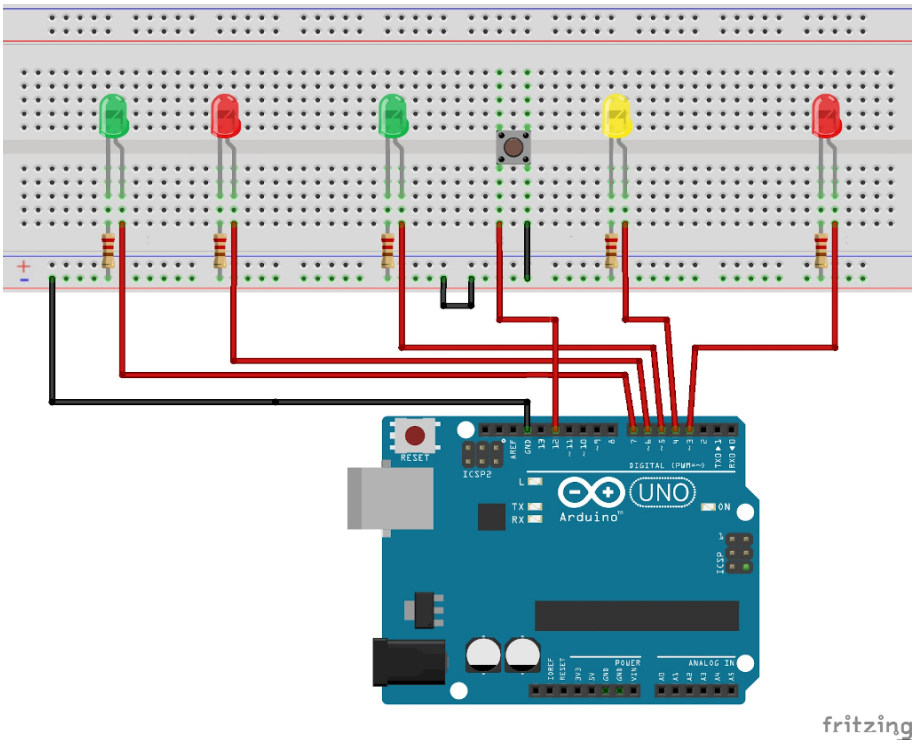
Eine Ampel mit Fußgängerampel soll mit einem Taster geschaltet werden. Wird der Taster gedrückt, schaltet die Fahrbahnampel auf rot und die Fußgängerampel auf grün.



Benötigte Bauteile:

- 2 rote LEDs
- 2 grüne LEDs
- 1 gelbe LED
- 5 Widerstände > 100 Ω
- Taster
- Leitungsdrähte

Baue die Schaltung auf.



Definiere zuerst die Variablen für die Zuordnung der LEDs und die Variable für den Taster.

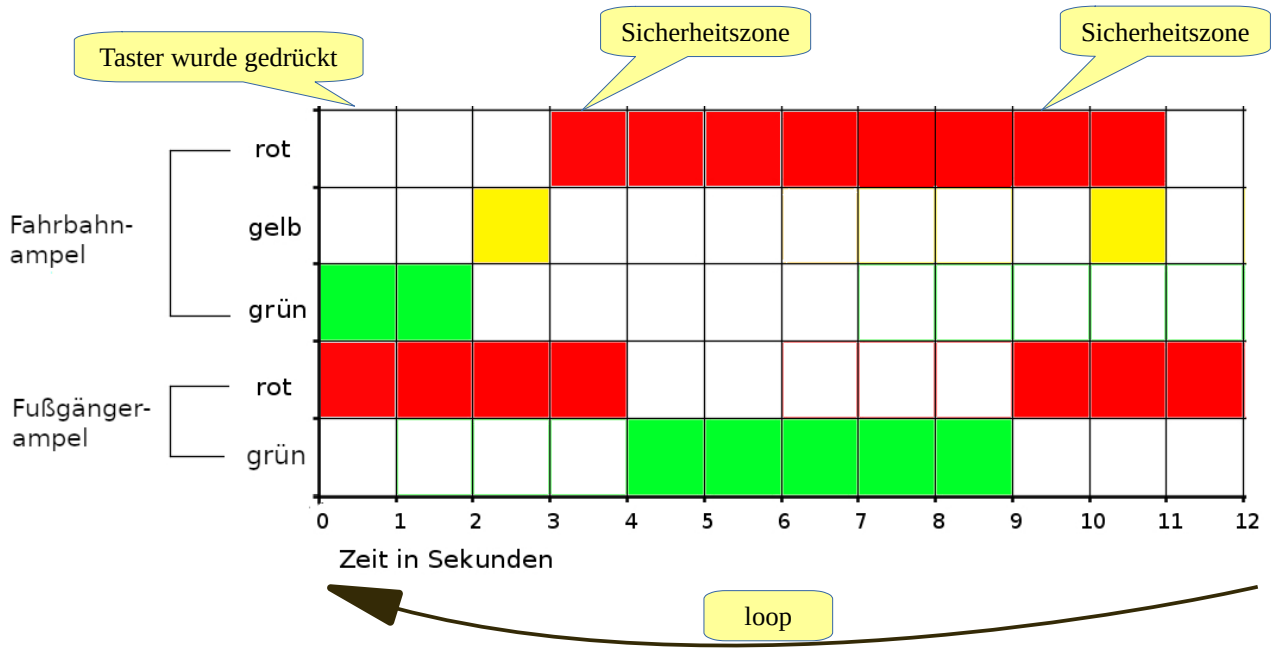
```
// Fahrbahnampel
int ROT = 3;
int GELB = 4;
int GRUEN = 5;

// Fußgängerampel
int F_ROT = 6;
int F_GRUEN = 7;
int TASTER = 12;
```

Für den Status des Taster wird ebenfalls eine Variable (TasterLesen) definiert: Der gelesene Wert (LOW/HIGH) entscheidet über die Schaltung der Ampeln.

```
int TasterLesen;
```

Im setup-Teil muss der `pinMode` des Tasters hinzufügt und der Vorwiderstand (`INPUT_PULLUP`) eingeschaltet werden.

```
void setup()
{
  pinMode(ROT, OUTPUT);
  pinMode(GELB, OUTPUT);
  pinMode(F_ROT, OUTPUT);
  pinMode(F_GRUEN, OUTPUT);

  // Taster
  pinMode(TASTER, INPUT_PULLUP);
}
```



Das Programm muss feststellen können, ob der Taster gedrückt wurde. Hierzu wird eine if-Abfrage benutzt.

Sie hat die Form:

```
// wenn die Bedingung erfüllt ist ...
if (Bedingung == Zustand)
{
   // Befehl ausführen
}
```

In diesem Fall der Zustand der Variable Tasterlesen abgefragt. Ist der Zustand LOW, wird die Ampel geschaltet.

Zunächst zeigt die Fußgängerampel rot und die Fahrbahnampel grün. Ein Druck auf den Taster unterbricht den Strom. Der Status des Tasters ist dann LOW.

Das wird mit `if` abgefragt:

```
void loop()
{
  // Ampel grün/Fußgängerampel rot
  digitalWrite(F_ROT, HIGH);
  digitalWrite(GRUEN, HIGH);

  // Zustand des Tasters lesen
  TasterLesen = digitalRead(TASTER);

  // LOW → Taster gedrückt
  if (TasterLesen == LOW)
  {
    // 1 s Pause
    delay(1000);

    // Fahrbahnampel grün aus
    digitalWrite(GRUEN, LOW);

    // Fahrbahnampel gelb an
    digitalWrite(GELB, HIGH);
    delay(1000);
    digitalWrite(GELB, LOW);

    // Fahrbahnampel rot an
    digitalWrite(ROT, HIGH);

    // 1 s Sicherheitszeit
    delay(1000);

    // Fußgängerampel auf grün schalten
    digitalWrite(F_ROT, LOW);
    digitalWrite(F_GRUEN, HIGH);
    delay(5000);
    // Fußgängerampel auf rot schalten
    digitalWrite(F_GRUEN, LOW);
```

```
    digitalWrite(F_ROT, HIGH);

    // 1 s Pause
    delay(1000);

    // Fahrbahnampel auf gelb und dann auf rot schalten
    digitalWrite(GELB, HIGH);
    delay(1000);
    digitalWrite(ROT, LOW);
    digitalWrite(GELB, LOW);
  }
}
```

Letzte Änderung: 16.08.22

Um ein Arduino-Programm zu schreiben, das einen Servomotor von der Position 0° langsam auf 90° innerhalb von 10 Sekunden bewegt, dann 10 Sekunden wartet und wieder auf 0° zurückfährt und nach weiteren 45 Sekunden die Bewegung wiederholt, können Sie die `Servo` -Bibliothek verwenden, die es Ihnen ermöglicht, einen Servomotor einfach zu steuern. Hier ist ein Beispielcode:

```
// Definieren Sie die Pins, an denen die LEDs angeschlossen sind
const int roteLED = 10;    // Rote LED an Pin 10
const int gelbeLED = 9;    // Gelbe LED an Pin 9
const int grueneLED = 8;   // Grüne LED an Pin 8

void setup() {
  // Konfigurieren Sie die LED-Pins als Ausgänge
  pinMode(roteLED, OUTPUT);
  pinMode(gelbeLED, OUTPUT);
  pinMode(grueneLED, OUTPUT);
}

void loop() {
  // Rote LED leuchtet für 5 Sekunden
  digitalWrite(roteLED, HIGH);
  delay(5000);

  // Gelbe LED leuchtet für 2 Sekunden (rote LED ist aus)
  digitalWrite(roteLED, LOW);
  digitalWrite(gelbeLED, HIGH);
  delay(2000);

  // Grüne LED leuchtet für 5 Sekunden (gelbe LED ist aus)
  digitalWrite(gelbeLED, LOW);
  digitalWrite(grueneLED, HIGH);
  delay(5000);

  // Gelbe LED leuchtet für 2 Sekunden (grüne LED ist aus)
  digitalWrite(grueneLED, LOW);
  digitalWrite(gelbeLED, HIGH);
  delay(2000);

  // Gelbe LED wird ausgeschaltet, und der Zyklus beginnt von vorn
  digitalWrite(gelbeLED, LOW);
}
```

```
#include <Servo.h>

Servo meinServo;  // Erstellen Sie ein Servo-Objekt
int pos = 0;      // Variable für die aktuelle Position

void setup() {
  meinServo.attach(9); // Der Servo ist am Pin 9 angeschlossen
}

void loop() {
  // Bewegen Sie den Servo langsam zur Position 90 Grad
  for (pos = 0; pos <= 90; pos += 1) { // Geht von 0 Grad zu 90 Grad
    meinServo.write(pos);              // Setzt die Position
    delay(111);                        // Wartet 111ms für jede Einheit,
um 10 Sekunden für 90 Grad zu erreichen
  }

  delay(10000); // Warten Sie 10 Sekunden bei 90 Grad

  // Bewegen Sie den Servo zurück zu 0 Grad
  for (pos = 90; pos >= 0; pos -= 1) { // Geht von 90 Grad zurück zu 0
Grad
    meinServo.write(pos);              // Setzt die Position
    delay(111);                        // Wartet 111ms für jede Einheit,
um 10 Sekunden für 90 Grad zu erreichen
  }

  delay(45000); // Warten Sie 45 Sekunden bei 0 Grad, bevor der Vorgang
wiederholt wird
}
```

Dieses Skript initialisiert drei Pins als Ausgänge für die LEDs. In der **`loop()`**-Funktion wird dann ein typischer Ampelzyklus simuliert, wobei die rote LED zuerst leuchtet, gefolgt von der gelben LED und dann der grünen LED. Zwischen den Phasen schaltet das Programm die gelbe LED für einen kurzen Zeitraum ein, um den Übergang zwischen Rot und Grün sowie zwischen Grün und Rot darzustellen. Die **`delay()`**-Funktion wird verwendet, um die Dauer jeder Phase festzulegen.
Bevor Sie diesen Code auf Ihrem Arduino ausführen, stellen Sie sicher, dass jede LED über einen geeigneten Vorwiderstand verfügt, um sie vor zu hohem Strom zu schützen, und dass die LEDs korrekt an die definierten Pins angeschlossen sind.