

## Tinkercad Circuits Level 2: Programmiere einen Arduino und lass eine LED Blinken

Mit „Tinkercad Circuits“ kannst du ganz einfach virtuelle elektrische Schaltungen bauen und dabei spielerisch die Welt der Elektronik entdecken! Du brauchst dafür nichts weiter als diese Anleitung und einen Computer oder ein Tablet mit Internetzugang.  
Für diese Anleitung solltest du bereits das **Level 1** gemacht haben, in dem du lernst wie du in Tinkercad Circuits Bauteile miteinander verbindest und eine Simulation startest.

### So geht's:

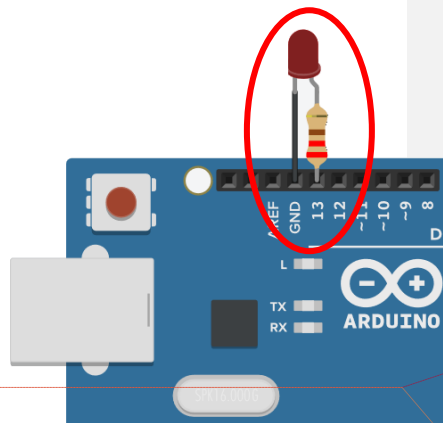
1. Öffne deinen Internet-Browser und logge dich auf [www.tinkercad.com](http://www.tinkercad.com) ein.

2. Klicke links auf **Circuits** und dann auf **Neuen Schaltkreis erstellen**

3. Los geht's! Ziehe eine **LED**, einen **Widerstand** und einen **Arduino** auf die Arbeitsfläche (siehe rechts). Mit der **R** - Taste kannst du ein angeklicktes Bauteil drehen

4. Verbinde die „Anode“ der LED (das lange Beinchen) mit dem Widerstand und die andere Seite des Widerstands mit „Pin 13“ am Arduino. Die Kathode der LED (das kurze Beinchen) wird mit einem Draht mit dem „GND“ Pin verbunden (siehe rechts).

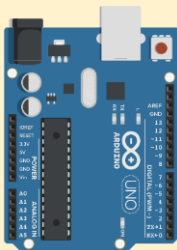
Tipp: Schau dir nochmal **Level 1** an, falls du Hilfe beim Verbinden der Bauteile brauchst.



**Kommentiert [KK1]:** Ich frage mich ob ein exkurs darüber was GND=ground=masse ist vs dem minus der batterie das letzte mal zu ausschweifend wird...

**Kommentiert [WF2R1]:**

### Was ist ein Arduino?





Ein Arduino ist ein kleiner Computer, den du selbst programmieren kannst. Mit ihm kannst du LEDs ein- und ausschalten, Druck- und Drehknöpfe lesen, Sensormessungen an deinen Computer schicken und vieles mehr. Die Bauteile schließt du an seinen Pin-Anschlüssen an. Der Arduino kann alles du brauchst um elektronische Musikinstrumente, nützliche Maschinen oder deine eigene Roboterarmee zu bauen!




5. Klicke auf den Widerstand  und ändere im kleinen Fenster rechts oben seinen Wert auf 220  $\Omega$ . Achte auf die richtige Einheit  $\Omega$  !

Widerstand	
Name	R1
Widerstand	220 $\Omega$

6. Klicke auf  **Simulation starten** . Die Led sollte nun Blinken!  
Klicke auf  **Simulation stoppen** bevor du weitermachst. Blinkt deine LED **nicht**?  
→ Überprüfe, ob alle Teile richtig miteinander verbunden sind!

Warum blinkt die LED eigentlich?  
Der Programm-Code macht's!

7. Klicke rechts oben auf  **Code** und wähle im Menü darunter „Text“ aus. Jetzt siehst du rechts den Code, der die LED blinken lässt. Der Code ist für den Arduino eine Art „Schritt-für-Schritt Anleitung“ – er denkt nicht darüber nach, sondern macht einfach genau das, was du ihm sagst!

Blocks
Blocks + Text
<b>Text</b>

### Arduino Code-Struktur

Ein Arduino-Programm besteht aus den zwei Hauptfunktionen **setup()** und **loop()**. Alle Befehle die zwischen den geschwungenen Klammern **{ }** stehen, werden vom Arduino nacheinander ausgeführt. Jeder Befehl **muss** mit einem Strichpunkt **;** enden.

```
1 void setup()
2 {
3   pinMode(13, OUTPUT);
4 }
5
6 void loop()
7 {
8   digitalWrite(13, HIGH);
9   delay(1000); // Wait for 1000 millisecond(s)
10  digitalWrite(13, LOW);
11  delay(1000); // Wait for 1000 millisecond(s)
12 }
```

Alles was in **setup()** steht, wird nach dem Einschaltung nur einmal ausgeführt.

Alles was in **loop()** steht wird danach endlos wiederholt .

Doppelte Schrägstriche **//** Kennzeichnen ein Kommentar.  
Alles was **danach** in derselben Zeile steht (in Orange), wird vom Arduino ignoriert.  
Du kannst **//** verwenden, um dir im Code Notizen zu machen!

**Kommentiert [KK3]:** Vermutlich eine gute idee code-snippets in einer monotype-schrift zu schreiben....

**Kommentiert [WF4R3]:** Das sehe ich auch so! Ich weiß zwar nicht genau warum, aber ich hab alle Snippets neu in „Consolas“ formatiert. Courier hat mir eigentlich besser gefallen.

8. Sehen wir uns den Code etwas genauer an! Der Befehl in Zeile 8:

```
digitalWrite(13, HIGH);
```

bewirkt, dass der Pin 13 (an den du die LED angeschlossen hast) „eingeschalten“ und mit Strom versorgt wird. **Die LED leuchtet!** Der Befehl in der nächsten Zeile:

```
delay(1000);
```

lässt den Arduino für 1000 Millisekunden (= 1 Sekunde) warten. Der Pin 13 wird weiter mit Strom versorgt und die LED leuchtet, sonst passiert nichts.

```
digitalWrite(13, LOW);
```

Schaltet den Strom auf Pin 13 wieder aus. Die LED leuchtet nicht mehr.

9. Jetzt ist es Zeit selbst zu programmieren! Ändere die Dauer der Wartezeiten z.B von `delay(1000)` auf `delay(200)`. Was passiert? Warum?

#### Bonus Level:



Mache folgenden Änderungen an deinem Schaltkreis und im Programm-Code (vergiss nicht die Simulation zu stoppen/starten):

- Verbinde den Widerstand mit Pin 11 anstelle von Pin 13. Blinkt die LED, wenn du die Simulation startest?
- Suche die Stellen im Programmcode, die du ändern musst, um die LED wieder zum Blinken zu bekommen. Tipp: *Du musst den Code an 2 Stellen ändern!*
- Versuche 2 LEDs an den Arduino anzuschließen, die Abwechselnd blinken. Du kannst ihnen auch unterschiedliche Farben geben!
- Du könntest mit deinem Wissen sogar eine Ampelschaltung nachbauen!

Im **techLAB** haben wir „echte“ Arduinos und viele elektronische Bauteile, mit denen du experimentieren kannst! Im techLAB gibt es aber noch viel mehr spannende Technologien zu entdecken. Arbeite mit 3D-Druckern, Laser-Cuttern, Stickmaschinen und mehr im **Technischen Museum Wien**. Die Öffnungszeiten des techLAB findest du [online](#). Der Eintritt ins Museum ist <19 Jahren gratis!

**Wir freuen uns auf deinen Besuch!**