



**DAS BASIC STARTER KIT TUTORIAL
FÜR UNO**

V1.0.19.7.24

Vorwort

Unser Unternehmen

Seit der Gründung in 2011 ist Eligo Inc. ein wachsendes Unternehmen im Technologie-Sektor, das sich der Forschung und Entwicklung sowie der Produktion und dem Marketing von Open-Source Hardware widmet. Mit Sitz in Shenzhen, dem Silicon Valley von China, sind wir zu einem über 150 Mitarbeiter großen Unternehmen mit einer Betriebsfläche von mehr als 10.763 Quadratmetern geworden.

Unsere Produkte gehen von DuPont Jumperkabeln über UNO R3 Entwicklungsboards bis hin zu kompletten Starter Kits, die für Anwender aller Art und Wissensstand geeignet sind, die die Arduino Programmierung lernen wollen. Außerdem verkaufen wir Zubehör für Raspberry Pi Mikrocontroller, wie zum Beispiel 2.8" TFT Touch Displays und STM32 MCUs. In Zukunft fokussieren wir uns auf die Entwicklung von 3D Druckern und passendem Zubehör, da wir an die aufstrebende Drucktechnologie glauben. Alle unsere Produkte erfüllen internationale Qualitätsstandards und sind in Ländern auf der ganzen Welt zugelassen.

Unsere Offizielle Website: <http://www.elegoo.com>

US Amazon storefront: <http://www.amazon.com/shops/A2WWHQ25ENKVJ1>

CA Amazon storefront: <http://www.amazon.ca/shops/A2WWHQ25ENKVJ1>

UK Amazon storefront: <http://www.amazon.co.uk/shops/AZF7WYXU5ZANW>

DE Amazon storefront: <http://www.amazon.de/shops/AZF7WYXU5ZANW>

FR Amazon storefront: <http://www.amazon.fr/shops/AZF7WYXU5ZANW>

ES Amazon storefront: <http://www.amazon.es/shops/AZF7WYXU5ZANW>

IT Amazon storefront: <http://www.amazon.it/shops/AZF7WYXU5ZANW>

Unsere Anleitung

Diese Anleitung richtet sich an Anfänger. Sie werden alle Grundinformationen erhalten und erfahren, wie man ein Arduino Controller Board, Sensoren und andere Komponenten benutzt und programmiert. Wenn Sie sich tiefergehend mit dem Thema Arduino beschäftigen wollen, empfehlen wir Ihnen das „*Arduino Cookbook*“ von Michael Margolis zu lesen.

Teile des Codes in dieser Anleitung wurden von Simon Monk geschrieben. Simon

Monk ist ein Autor vieler Bücher im Bereich von Open-Source Hardware. Seine Bücher kann man ebenfalls auf Amazon erwerben: „*Programming Arduino*“, „*30 Arduino Projects for the Evil Genius*“ und „*Programming the Raspberry Pi*“ sind ein paar Beispiele seiner Arbeit.

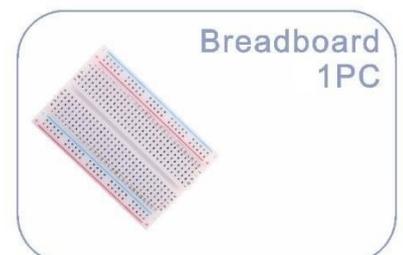
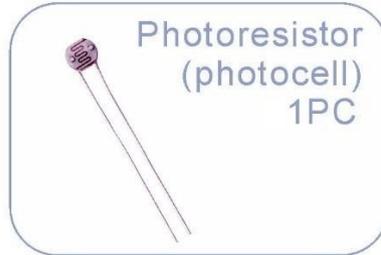
Kundendienst

Als ein andauerndes und schnell wachsendes Unternehmen geben wir unser Bestes, um Ihnen exzellente Produkte und guten Service zu bieten. Wir versuchen alle Ihre Erwartungen zu erfüllen und sollte es mal ein Problem geben, können Sie uns einfach erreichen, indem Sie uns eine kurze Nachricht an service@elegoo.com oder EUservice@elegoo.com schicken. Wir freuen uns von Ihnen zu hören und schätzen jegliche Kritik und Verbesserungsvorschläge.

Wir antworten prompt auf jede Nachricht, die wir erhalten. Unsere erfahrenen Entwickler melden sich bei Ihnen innerhalb von 12 Stunden bzw. 24 Stunden während der Feriensaison.

Packing list

www.elegoo.com



Contact us : service@elegoo.com

Inhalt

Lektion 0: Installation des IDE	6
Lektion 1: Bibliotheken einbinden / Seriellen Monitor öffnen	17
Lektion 2: Blink	26
Lektion 3: LEDs	37
Lektion 4: RGB LEDs	44
Lektion 5: Digitale Eingänge	53
Lektion 6: Aktiver Buzzer	58
Lektion 7 Neigungssensor	62
Lektion 8 Acht LEDs per 74HC595 ansteuern	66
Lektion 9 Der Serielle Monitor	73
Lektion 10 Fotозelle	79

Lektion 0: Installation des IDE

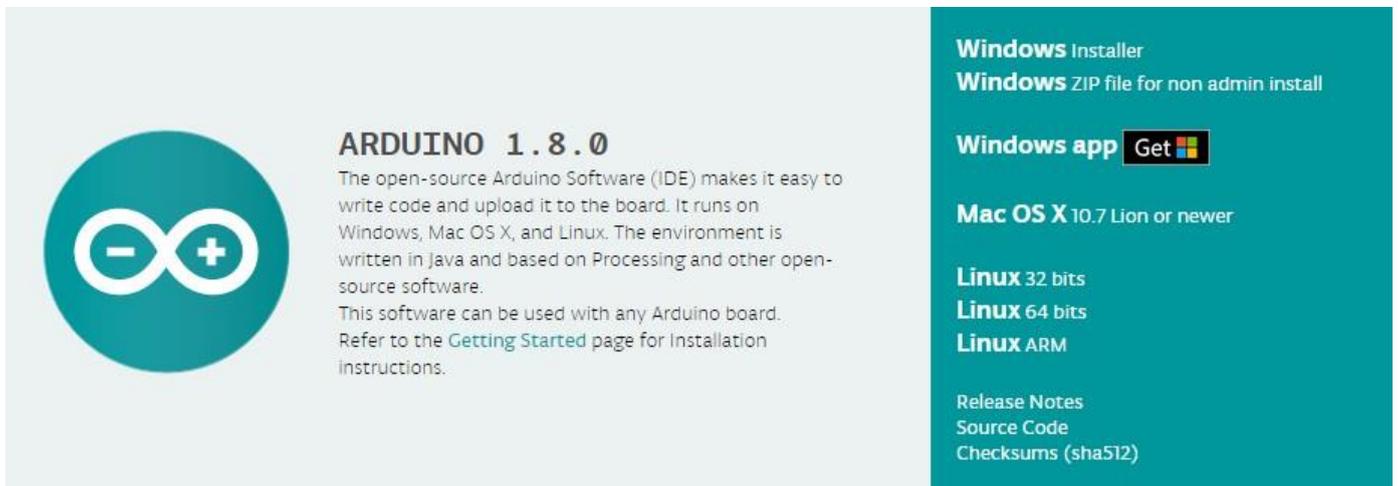
Einführung

Das Arduino IDE („Integrated Development Environment“ = Integrierte Entwicklungsumgebung) ist die Software der Arduino Plattform, mit der man programmiert.

In dieser Lektion werden Sie lernen, wie man den Computer zur Verwendung mit Arduino einrichtet, um die anschließenden Lektionen erfolgreich durchführen zu können.

Die Arduino Software, die Sie zum Programmieren des Arduino benutzen werden, ist für Windows, Mac und Linux verfügbar. Die Installation ist für jede der drei Plattformen unterschiedlich und einige Dinge müssen manuell eingestellt werden, um die Software einzurichten.

Schritt 1: Gehen Sie auf die Seite <https://www.arduino.cc/en/Main/Software> und suchen Sie diese Anzeige:



ARDUINO 1.8.0

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer
Windows ZIP file for non admin install

Windows app [Get](#)

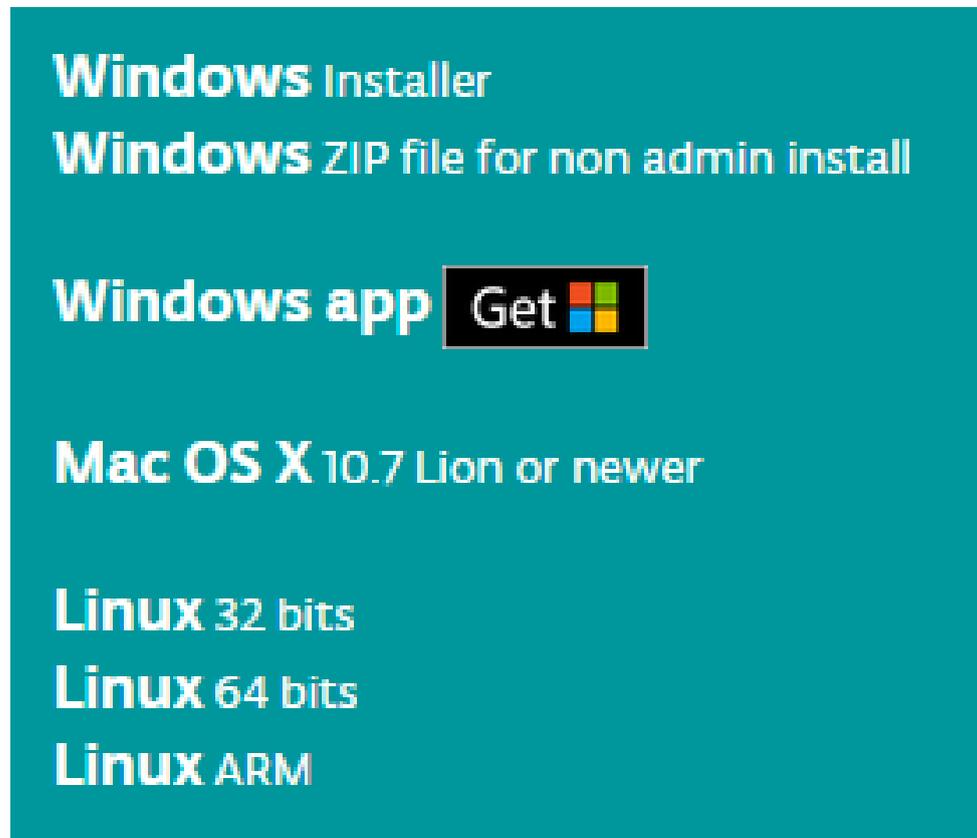
Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

[Release Notes](#)
[Source Code](#)
[Checksums \(sha512\)](#)

Auf dieser Website finden Sie immer die aktuellste Version der Software, daher kann sich die Version im Bild von der Ihnen angezeigten unterscheiden.

Schritt 2 : Laden Sie die für Ihr Betriebssystem vorhergesehene Variante der Entwicklungsumgebung herunter. Als Beispiel nehmen wir die Windows Variante:



Klicken Sie auf *Windows Installer*.

Support the Arduino Software

Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.

SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED **8,808,272** TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

\$3 \$5 \$10 \$25 \$50 OTHER

JUST DOWNLOAD **CONTRIBUTE & DOWNLOAD**

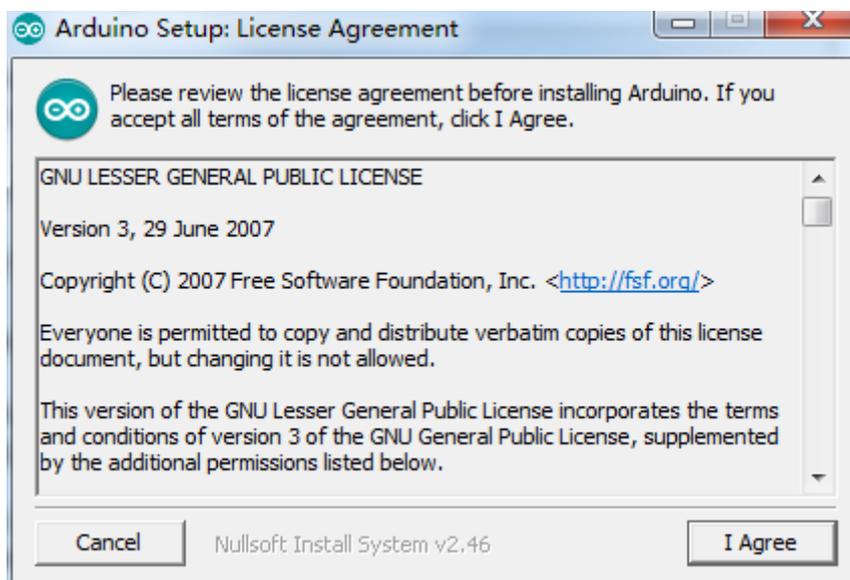
Klicken Sie auf *JUST DOWNLOAD*.

Falls es Ihnen nicht möglich ist die neueste Version von der Arduino Website herunterzuladen, können Sie auch die von uns bereitgestellte Version 1.8.0 verwenden, die Sie im Materialpaket finden. Dies ist die aktuellste Version zum Zeitpunkt der Veröffentlichung dieser Anleitung.

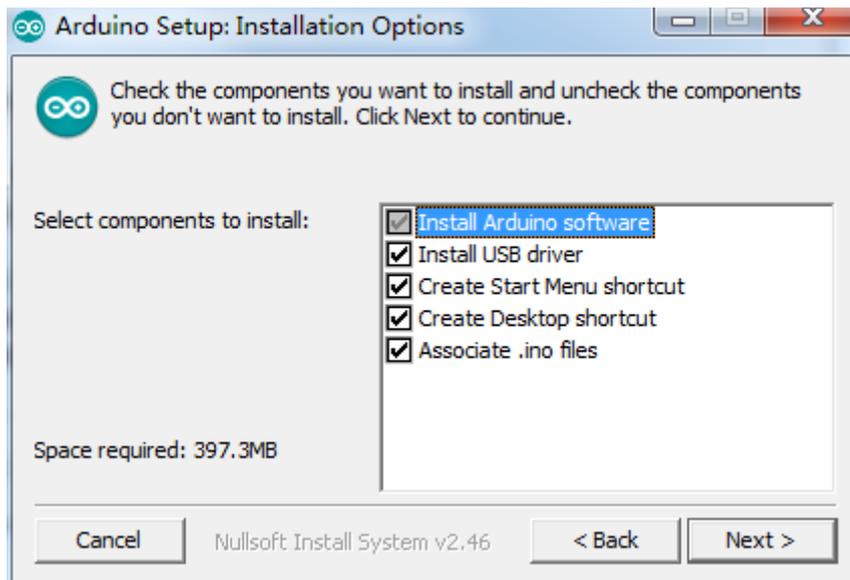
-  `arduino-1.8.0-linux32.tar.xz`
-  `arduino-1.8.0-linux64.tar.xz`
-  `arduino-1.8.0-macosx.zip`
-  `arduino-1.8.0-windows.exe`
-  `arduino-1.8.0-windows.zip`

Arduino IDE installieren (Windows)

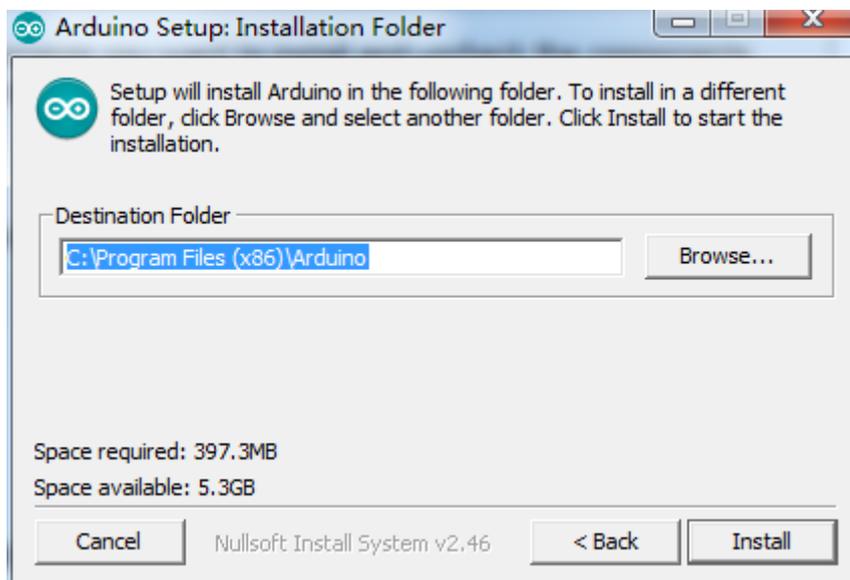
Installation mit dem Installationspaket (.exe) - *empfohlen* -



Klicken Sie auf *I Agree*, um fortzufahren.

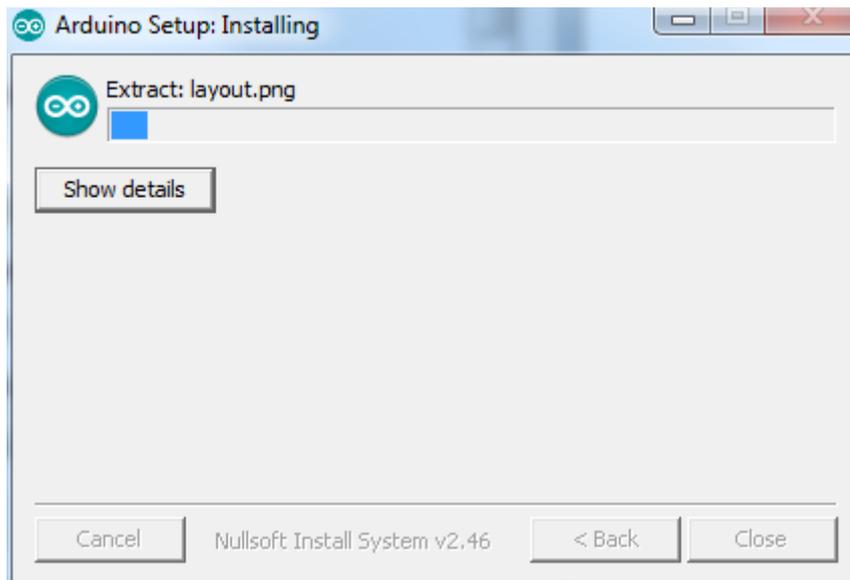


Klicken Sie auf *Next*.

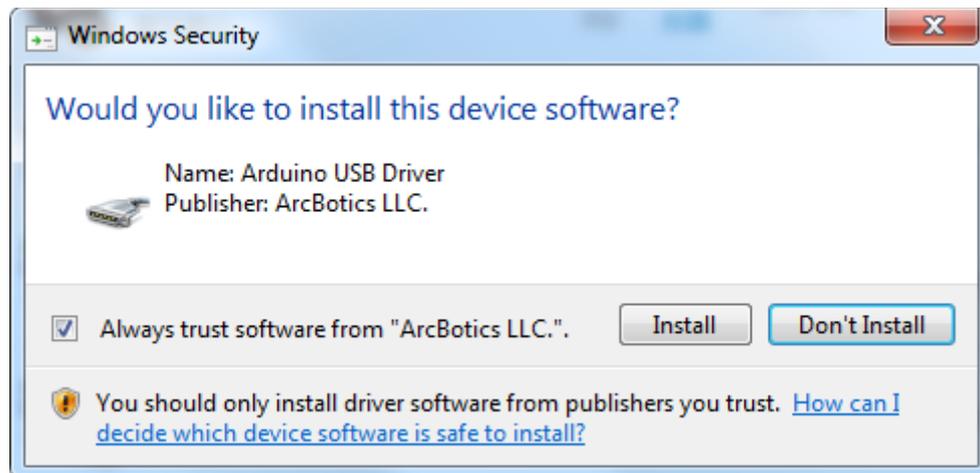


Fortgeschrittene können *Browse...* anklicken, um den Installationspfad zu ändern. Sie können diesen der Einfachheit halber aber auch einfach so belassen, wie er standardmäßig eingetragen ist.

Klicken Sie auf *Install*, um die Installation zu starten.



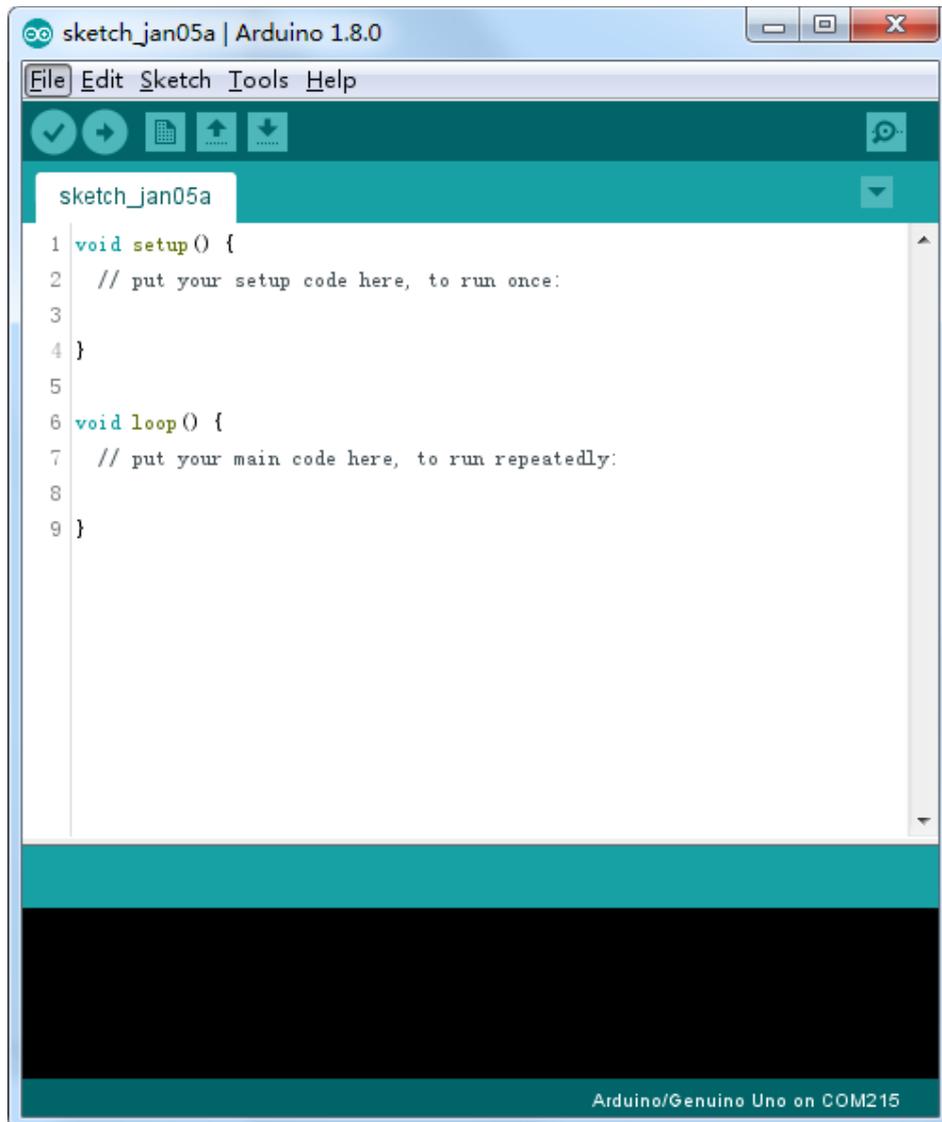
Zum Schluss erscheint diese Meldung. Klicken Sie auf *Installieren* und Sie sind fertig.



Schließlich erscheint folgendes Symbol auf dem Desktop.



Doppelklicken Sie auf das Symbol zum Starten der Arduino Entwicklungsumgebung.

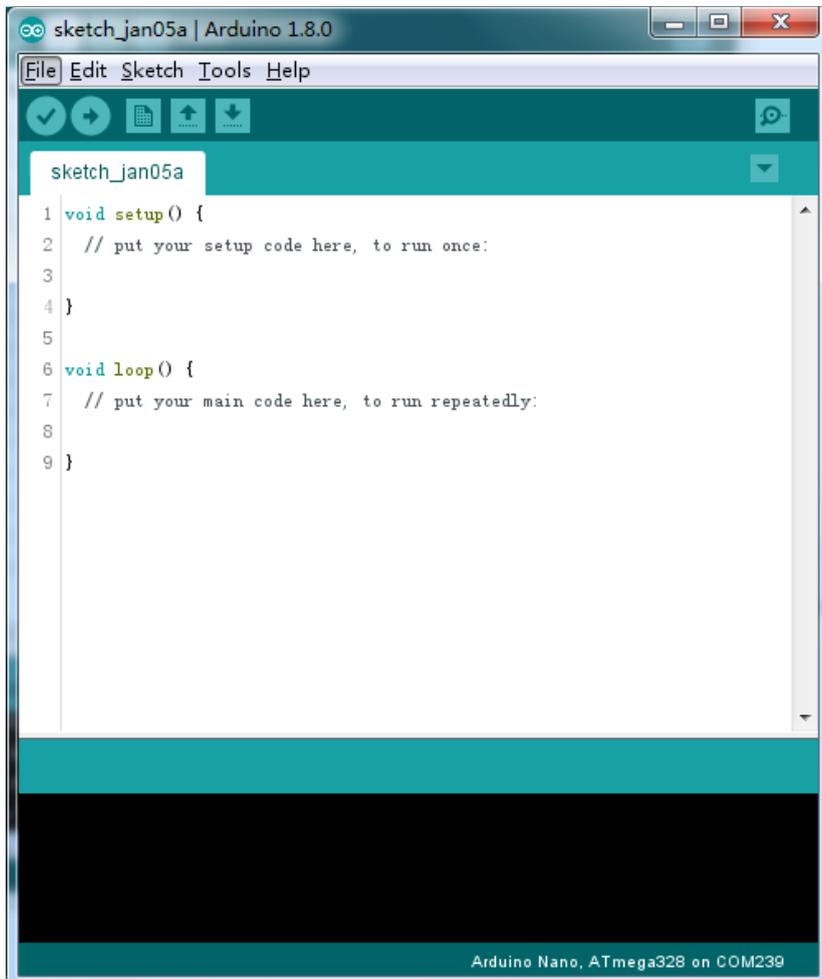
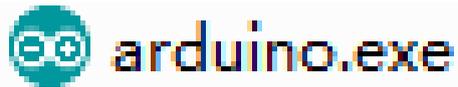
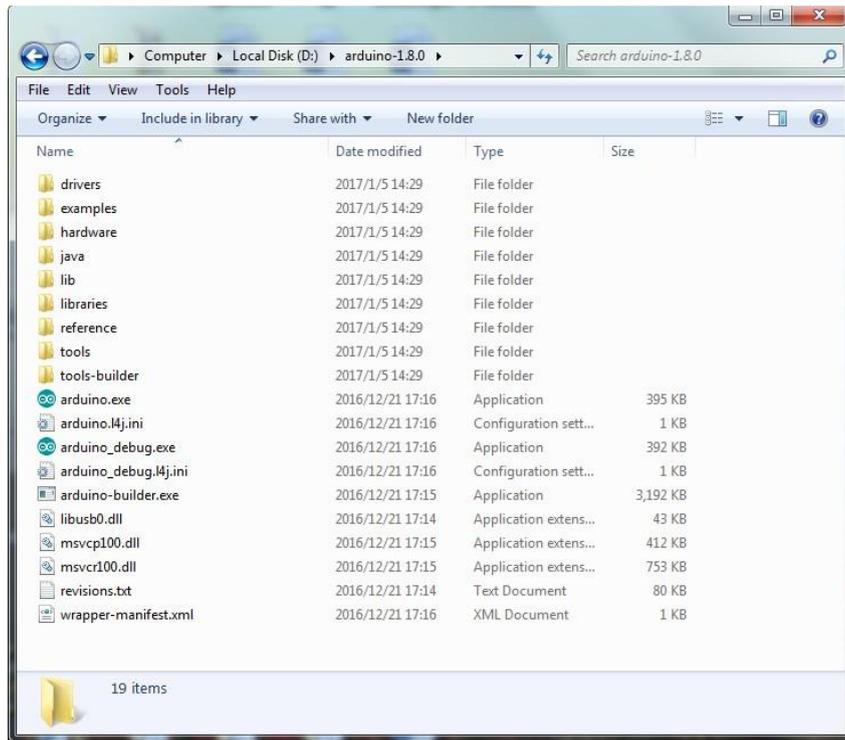


Sie haben die Installation nun erfolgreich abgeschlossen und können zu Lektion 1 gehen. Nachfolgend werden Installationsmethoden für andere Betriebssysteme beschrieben, die Sie überspringen können.

Installation mit dem ZIP-Archiv (.zip) - Für Fortgeschrittene -



Entpacken Sie das Archiv in einen beliebigen Ordner, in dem Sie das Programm installiert haben wollen. Dies ist Ihr „*Arduino-Ordner*“. Doppelklicken Sie auf die *arduino.exe* im Arduino-Ordner, um die Entwicklungsumgebung zu starten.



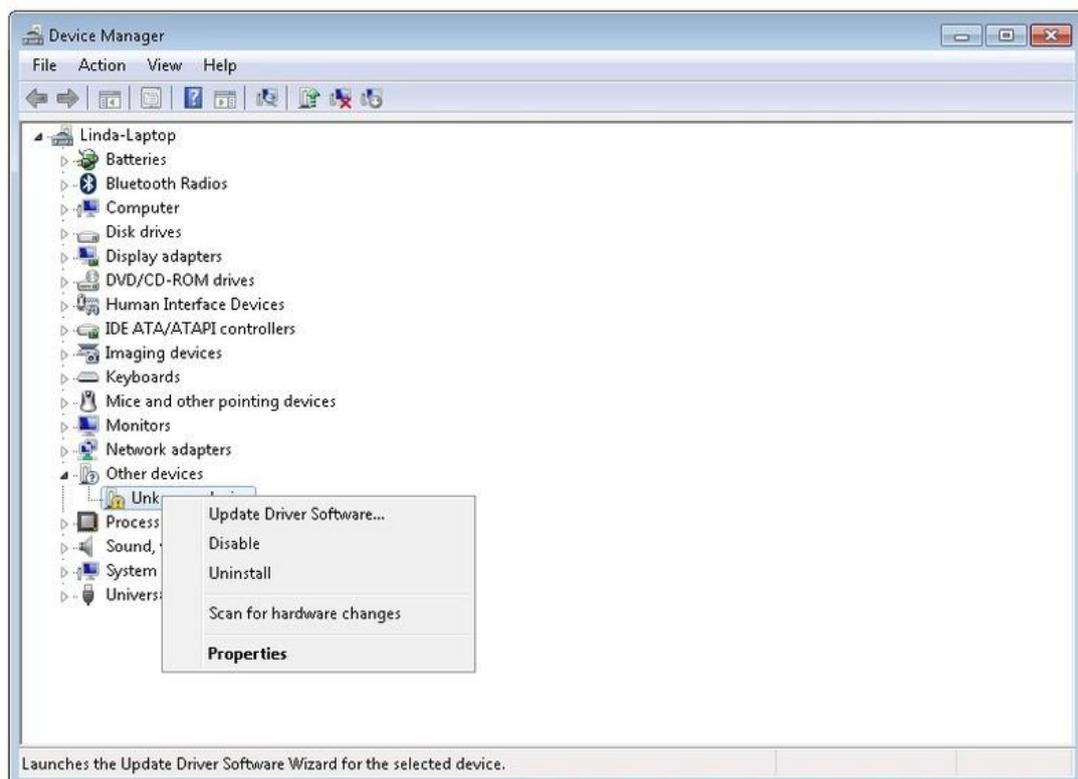
Die Installation war einfach, doch bei dieser Methode müssen die Treiber manuell installiert werden.

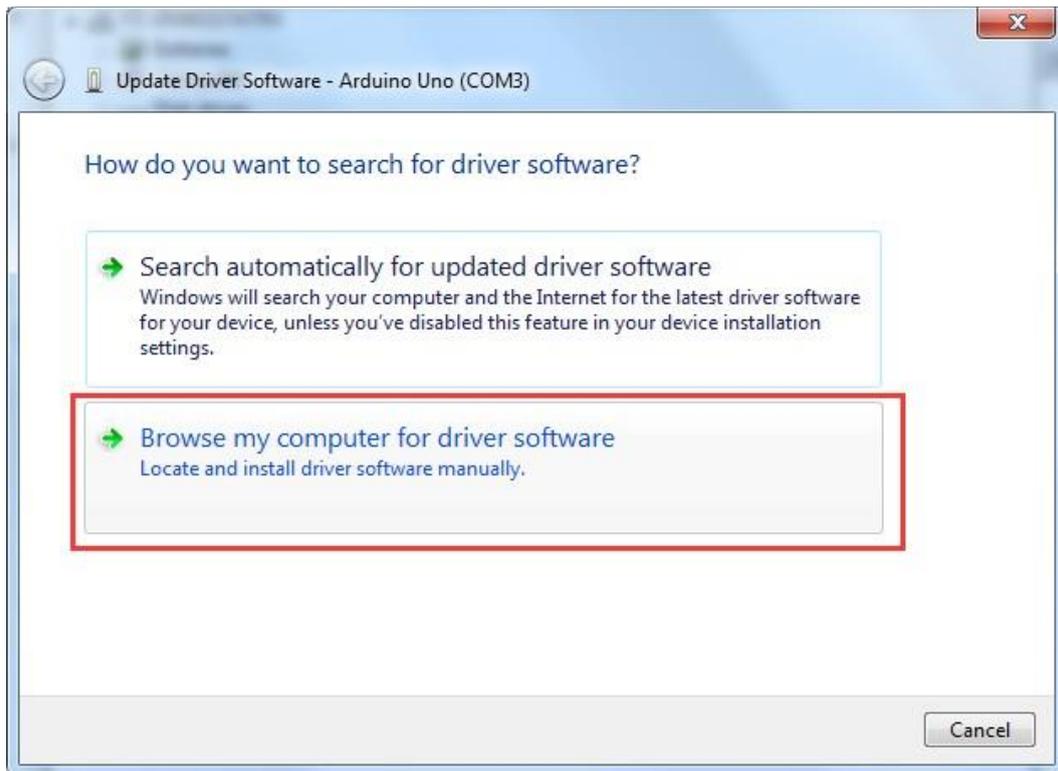
Der Arduino Ordner enthält neben der Entwicklungsumgebung auch die nötigen Treiber, die es ermöglichen den Arduino mit Ihrem Computer via USB-Kabel zu verbinden. Wir schließen die Entwicklungsumgebung wieder und installieren zuerst die USB Treiber.

Verbinden Sie das beigelegte USB-Kabel mit dem Arduino und das andere Ende mit dem Computer. Die Power LED des Arduinos leuchtet auf und möglicherweise erhalten Sie eine Nachricht von Windows, dass ein unbekanntes Gerät verbunden wurde. Ignorieren Sie diese Nachricht und schließen alle Anfragen von Windows einen Treiber automatisch installieren zu wollen.

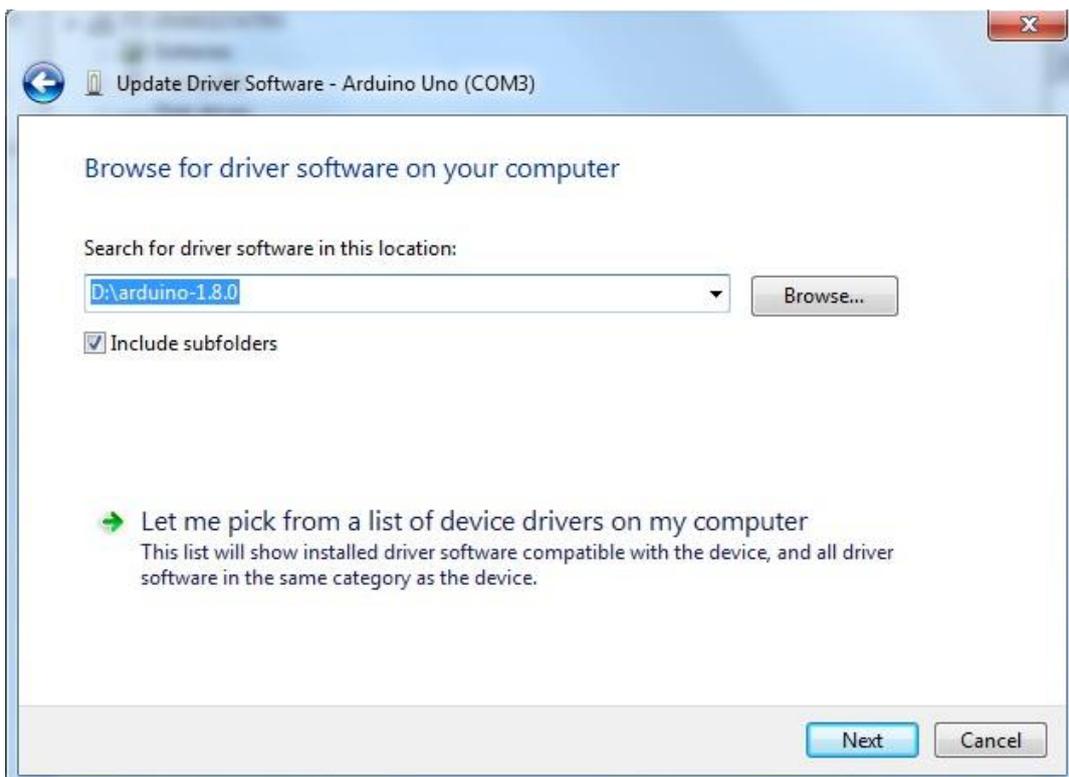
Die sicherste Methode, die USB-Treiber zu installieren, ist über den Geräte-Manager. Dieser kann abhängig von der Windows Version auf unterschiedliche Weisen geöffnet werden. Suchen Sie einfach im Startmenü nach „Geräte-Manager“ und Sie sollten fündig werden. Alternativ können Sie die Systemsteuerung öffnen und klicken dort auf *Hardware und Sound*. Unter dem Punkt *Geräte und Drucker* finden Sie ebenfalls den Eintrag zum Geräte-Manager. Ist der Geräte-Manager geöffnet, sehen Sie eine ähnliche Ansicht, wie die untere.

Unter „Andere Geräte“ sollten Sie einen Eintrag mit einer gelben Warn-Markierung sehen. Dies ist Ihr Arduino.





Machen Sie einen Rechtsklick auf das Unbekannte Gerät und wählen Sie die erste Option (*Treibersoftware aktualisieren...*) aus. Sie haben dann die Optionen „Automatisch nach aktueller Treibersoftware suchen“ und „Auf dem Computer nach Treibersoftware suchen“. Klicken Sie auf letzteres und navigieren Sie anschließend zu Ihrem Arduino-Ordner und dann zu dem darin liegenden Ordner *drivers* (Im Beispiel ist es `D:\arduino-1.8.0\drivers`)



Klicken Sie auf *Weiter*. Wenn Sie eine Sicherheitswarnung erhalten, erlauben Sie die Installation des Treibers. Sobald der Treiber erfolgreich installiert wurde, erhalten Sie eine Bestätigung.



Sie haben die Installation nun erfolgreich abgeschlossen und können zu Lektion 1 gehen. Nachfolgend werden Installationsmethoden für andere Betriebssysteme beschrieben, die Sie überspringen können.

Arduino IDE Installieren (Mac OS X)

 **arduino-1.8.0-macosx.zip**

Laden Sie das Archiv herunter und entpacken Sie es anschließend. Doppelklicken Sie auf die *Arduino.app* Datei im entpackten Ordner, um die Entwicklungsumgebung zu starten. Falls Sie eine Anfrage bekommen, ob *Java Runtime Library* installiert werden soll, bestätigen Sie dies. Wenn die Installation abgeschlossen ist, können Sie die Entwicklungsumgebung jederzeit durch einen Doppelklick auf die *Arduino.app* öffnen.

Arduino IDE installieren (Linux) - Fortgeschritten -

 [arduino-1.8.0-linux32.tar.xz](#)

 [arduino-1.8.0-linux64.tar.xz](#)

Zuerst entpacken Sie das heruntergeladene Archiv (x32 ist für 32bit Betriebssysteme und x64 für 64bit). Sie müssen das Paket dann mit dem *make install* Befehl für Ihr System kompilieren. Dies ist kompliziert und erfordert erweiterte Systemkenntnisse. Bei weiteren Fragen lesen Sie sich bitte die *README* Datei im Archiv durch oder suchen im Internet unter Angabe Ihres Betriebssystems und „*Arduino Installieren*“ nach Hilfe.

Wenn Sie Ubuntu oder ein auf Ubuntu basierendes Linux Derivat benutzen, gibt es eine einfache alternative Möglichkeit die *Arduino IDE* über das Softwarecenter zu installieren. Öffnen Sie dazu das Software Center und geben im Suchfeld „*Arduino*“ ein und klicken anschließend auf *Installieren*.

TIPP: Wenn Sie Probleme bei der Treiberinstallation unter Linux haben, lesen Sie sich bitte das Dokument „*UNO R3, MEGA, NANO DRIVER FAQ*“ durch.

 [UNO R3, MEGA, NANO DRIVER FAQ](#)

Lektion 1: Bibliotheken einbinden / Seriellen Monitor öffnen

Zusätzliche Arduino Bibliotheken einbinden

Wenn Sie mit der Arduino Entwicklungsumgebung und dessen Funktionen vertraut sind, möchten Sie möglicherweise die Möglichkeiten Ihres Arduinos mit zusätzlichen Bibliotheken erweitern.

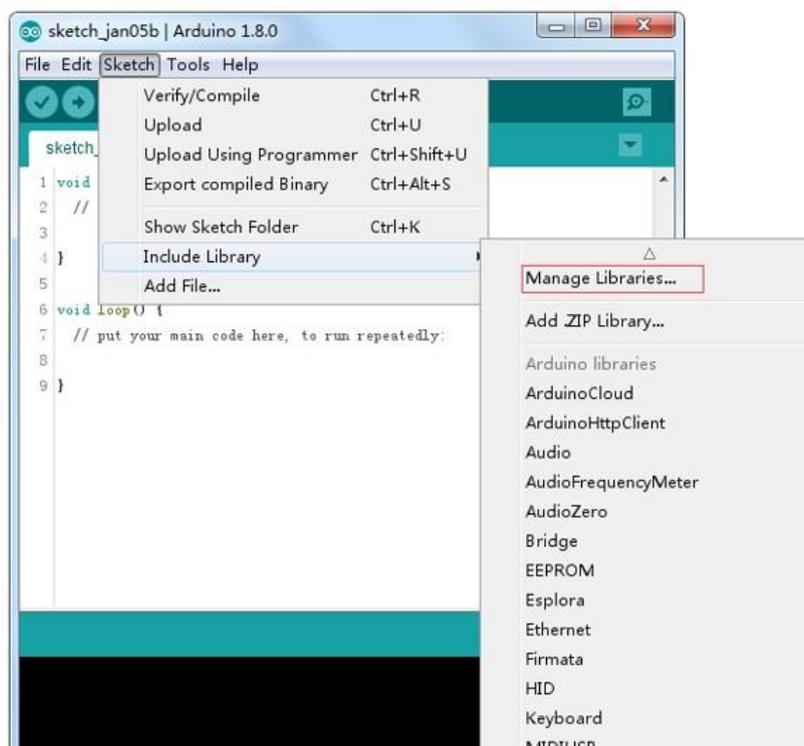
Was sind Bibliotheken?

Bibliotheken sind Sammlungen von vorprogrammiertem Code, die es enorm einfach für Sie machen beispielsweise einen Sensor, ein Display oder ein Modul einzubinden. Zum Beispiel vereinfacht die vorinstallierte *LiquidCrystal* Bibliothek die Kommunikation zu LCD-Zeichenanzeigemodulen. Es gibt hunderte von zusätzlichen Bibliotheken für Arduino im Internet, die Sie sich einfach herunterladen können. Die mit der Arduino IDE mitkommenden Bibliotheken und einige zusätzliche Bibliotheken sehen Sie später zum Vergleich. Um zusätzliche Bibliotheken benutzen zu können, müssen diese erst installiert werden.

Wie installiert man eine Bibliothek?

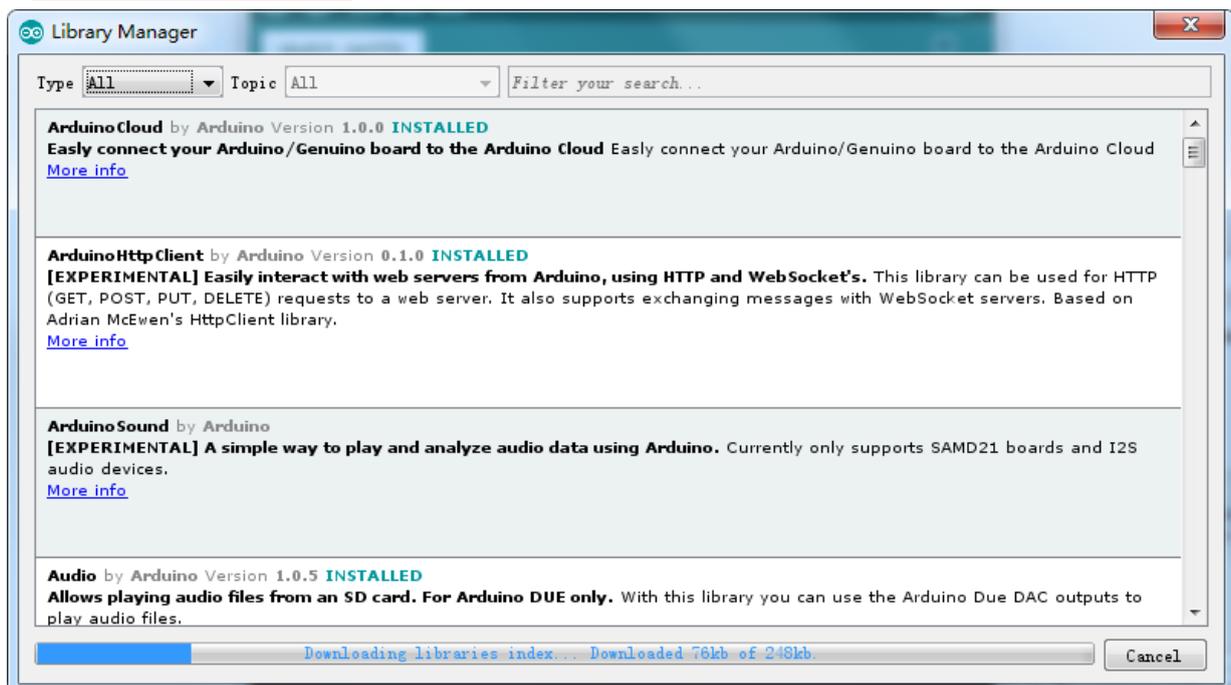
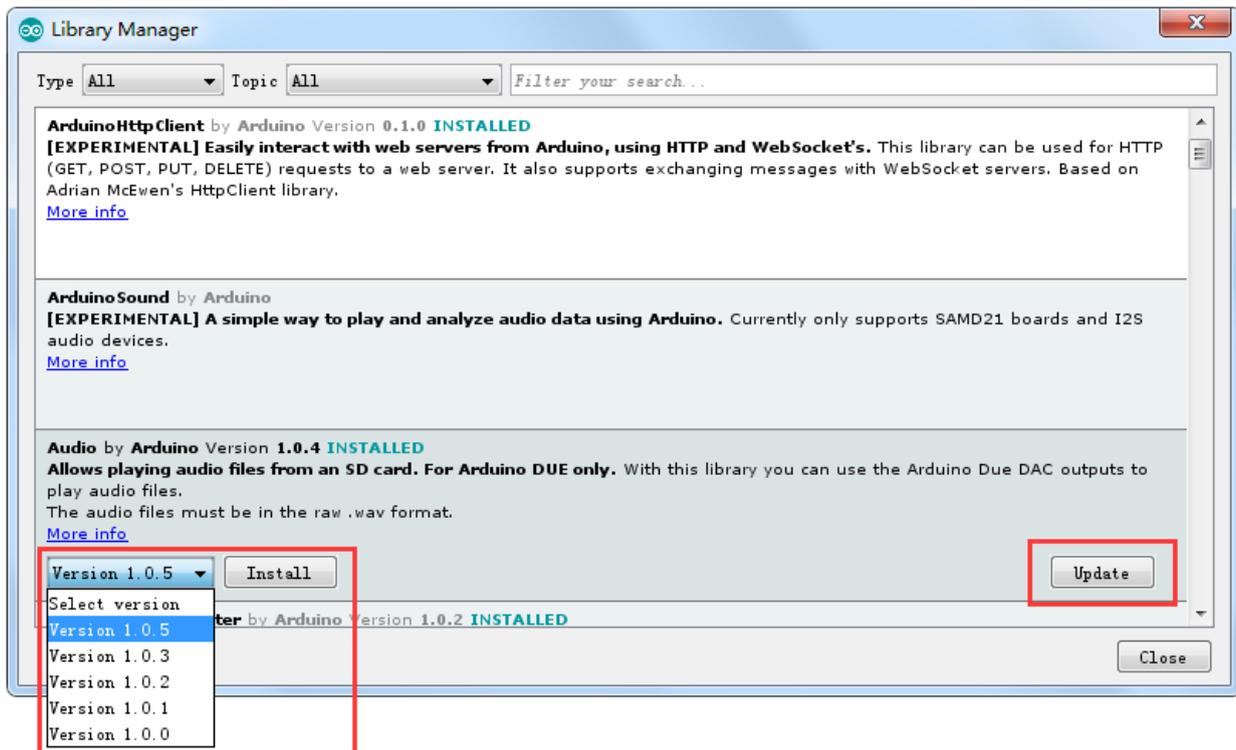
Mit Hilfe des Bibliotheksverwalters

Um eine neue Bibliothek in Ihre Arduino IDE einzubinden, können Sie den *Bibliotheksverwalter* nutzen (verfügbar seit Arduino IDE 1.8.0). Öffnen Sie die IDE und klicken unter *Sketch > Bibliothek einbinden* auf „Bibliotheken verwalten“.



Dann wird sich der Bibliotheksverwalter öffnen und Sie finden eine Liste voller Bibliotheken, die bereits installiert sind oder heruntergeladen werden können. Als Beispiel werden wir die Bibliothek *Audio* installieren. Scrollen Sie die Liste herunter und suchen Sie die Bibliothek. Dann wählen Sie die Version aus, die Sie installieren wollen. Manchmal gibt es nur eine verfügbare Version und es lässt sich keine Version auswählen. Keine Sorge: Das ist normal.

Manchmal müssen Sie ein bisschen geduldig sein, da es sehr viele Bibliotheken gibt. Lassen Sie die Software die Liste aktualisieren und suchen Sie dann den Eintrag.



Klicken Sie zum Schluss auf *Installieren* und warten Sie, bis die Bibliothek installiert ist. Das Herunterladen kann abhängig von Ihrer Internetverbindungsgeschwindigkeit etwas dauern. Sobald die Installation abgeschlossen ist, sollte die Bibliothek im Verwalter den Zusatz „*Installiert*“ haben. Danach können Sie den Bibliotheksverwalter wieder schließen.

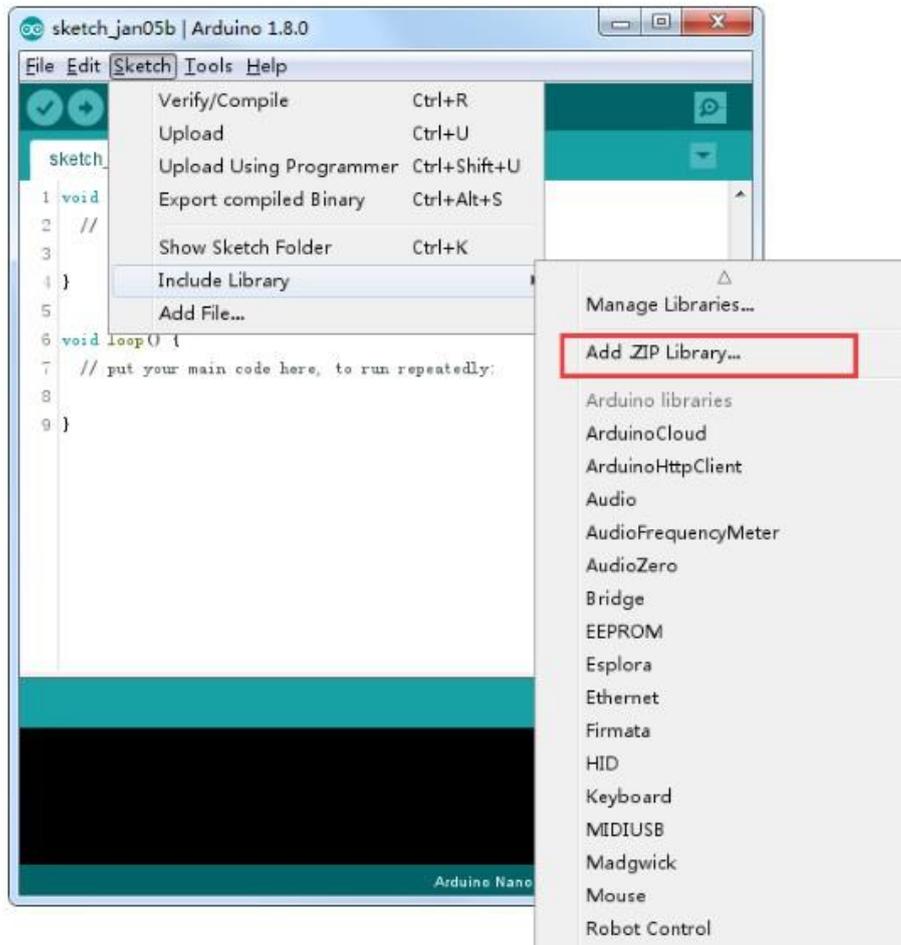


Jetzt können Sie die Bibliothek auch schon über das Einbindungsmenü hinzufügen. Wenn Sie Ihre eigene Bibliothek im Bibliotheksverwalter hinzufügen möchten, erstellen Sie einen Eintrag auf der [Github](#)-Seite von Arduino.

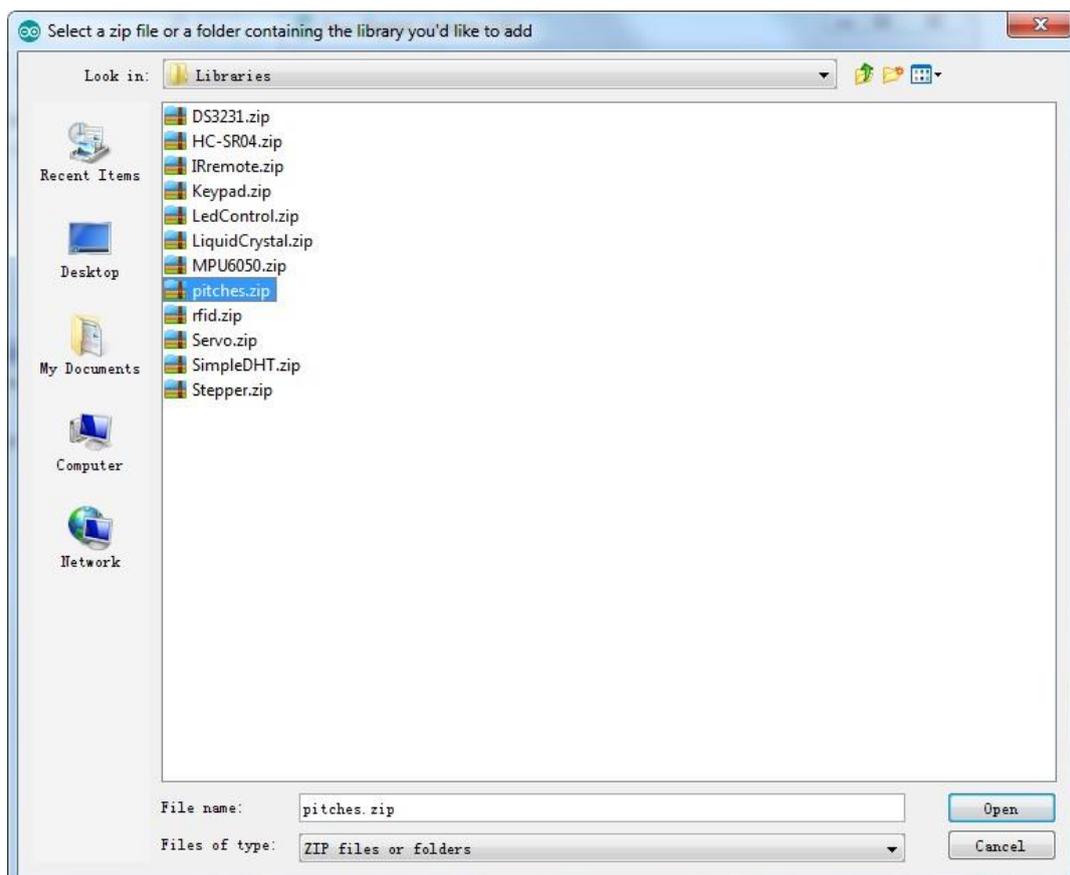
Eine .zip Bibliothek einbinden

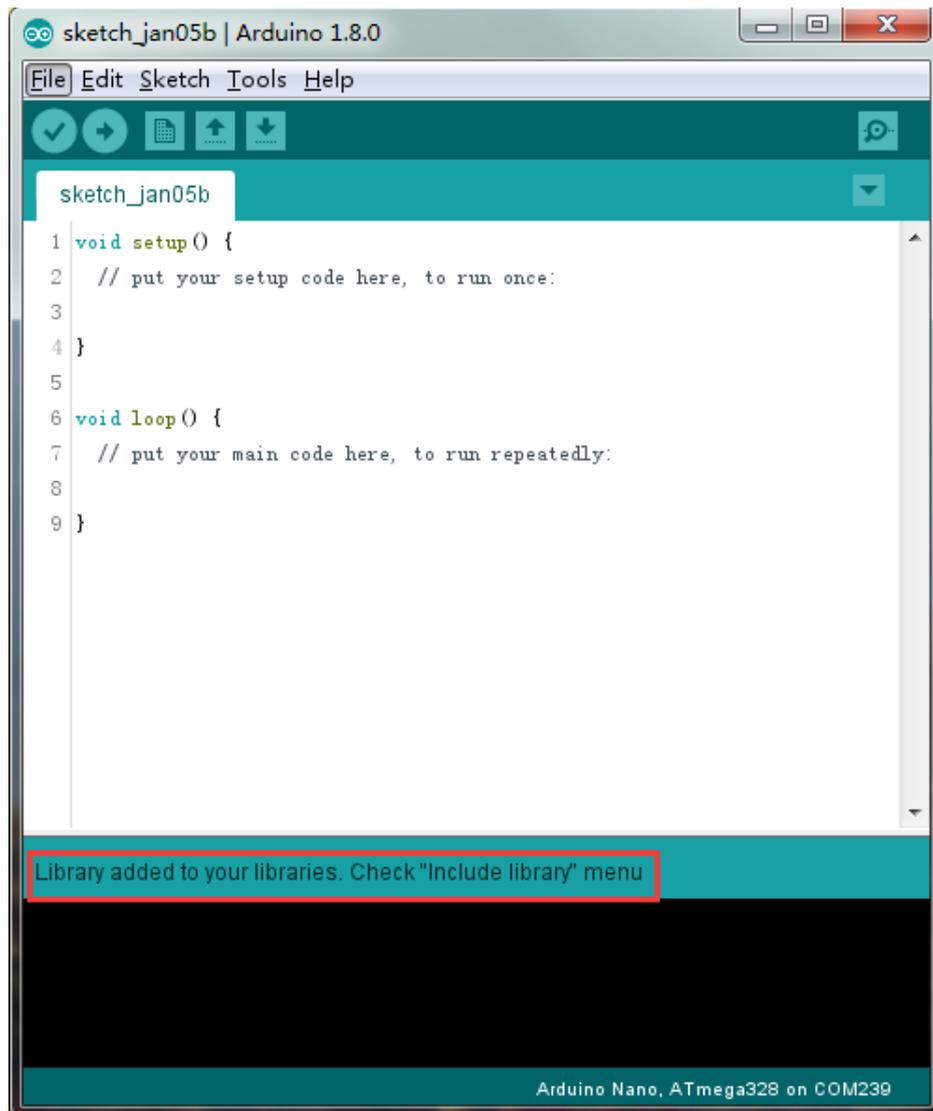
Weitere Bibliotheken werden im Internet häufig als ZIP Dateien zum Download angeboten. Der Name des ZIP Archivs ist meist der Name der Bibliothek. Im Archiv befindet sich eine *.cpp* Datei, eine *.h* Datei und oft eine *keywords.txt* Datei und ein *examples* Ordner mit Code-Beispielen und andere Dateien, die von der Bibliothek benötigt werden. Seit Version 1.0.5 können Drittanbieter-Bibliotheken in der Arduino IDE eingebunden werden. Entpacken Sie die ZIP Dateien der Bibliotheken nicht selber, sondern belassen Sie sie so wie sie sind. Die IDE wird das Archiv automatisch entpacken.

Navigieren Sie in der IDE zu *Sketch > Bibliothek einbinden* und wählen Sie „*ZIP Bibliothek hinzufügen...*“.



Sie werden dazu aufgefordert, die ZIP-Datei auszuwählen, die Sie hinzufügen wollen.
Öffnen Sie die heruntergeladene .ZIP-Bibliothek.





Gehen Sie zurück zum Menü *Sketch > Bibliothek einbinden*. Sie sollten Ihre soeben hinzugefügte Bibliothek unten im Drop-Down-Menü wiederfinden. Die Bibliothek kann nun in Ihrem Sketch / Projekt eingebunden werden. Die ZIP-Bibliothek wurde automatisch zu den anderen Bibliotheken in Ihrem Arduino Projektordner hinzugefügt.

Info: Die neue Bibliothek kann direkt in Ihren Sketch eingebunden werden. Die Beispieldateien der neuen Bibliothek sind jedoch erst nach einem Neustart der IDE verfügbar.

Dies sind beiden geläufigsten Wege, um Bibliotheken hinzuzufügen. Bei Mac und Linux Systemen funktioniert es auf die gleiche Art. Die untenstehende manuelle Installation wird nur selten gebraucht und falls Sie diese nicht benötigen, können Sie sie überspringen.

Manuelle Installation

Beenden Sie zuerst die Arduino Software, bevor Sie die Bibliothek manuell installieren. Dann entpacken Sie das ZIP-Archiv der Bibliothek. Wenn Sie zum Beispiel eine Bibliothek namens "ArduinoParty" heruntergeladen haben, entpacken Sie die *ArduinoParty.zip* Datei. Der entpackte Ordner sollte einen Ordner namens *ArduinoParty* enthalten, in welchem sich wiederum Dateien wie *ArduinoParty.cpp* und *ArduinoParty.h* befinden sollten. Wenn die *.cpp* und *.h* Dateien nicht in einem separaten Ordner liegen, müssen Sie einen erstellen. In diesem Fall würden Sie einen Ordner namens „*ArduinoParty*“ erstellen und verschieben alle Dateien, die im Zip-Archiv waren, in diesen Ordner (wie *ArduinoParty.cpp* und *ArduinoParty.h*). Verschieben Sie den *ArduinoParty* Ordner nun in Ihren Bibliotheksordner. Unter Windows und Mac ist dies standardmäßig unter *Dokumente\Arduino\libraries* . Unter Linux ist es ebenfalls *libraries* Ordner in Ihrem Projektordner.

Ihr Arduino Bibliotheks Ordner sollte nun so aussehen (unter Windows):

Dokumente\Arduino\libraries\ArduinoParty\ArduinoParty.cpp

Dokumente\Arduino\libraries\ArduinoParty\ArduinoParty.h

Dokumente\Arduino\libraries\ArduinoParty\examples

oder so (unter Mac und Linux):

Dokumente/Arduino/libraries/ArduinoParty/ArduinoParty.cpp

Dokumente/Arduino/libraries/ArduinoParty/ArduinoParty.h

Dokumente/Arduino/libraries/ArduinoParty/examples

....

Es können mehr Dateien als nur die *.cpp* and *.h* Dateien vorhanden sein, gehen Sie nur sicher, dass sich alle Dateien in Ihrem Ordner befinden. Die Bibliothek wird nicht funktionieren, wenn sich die *.cpp* und *.h* Dateien direkt im *libraries* Ordner oder in einem weiteren Unterordner befinden. Zum Beispiel:

Dokumente\Arduino\libraries\ArduinoParty.cpp und

Dokumente\Arduino\libraries\ArduinoParty\ArduinoParty\ArduinoParty.cpp

würden nicht funktionieren.

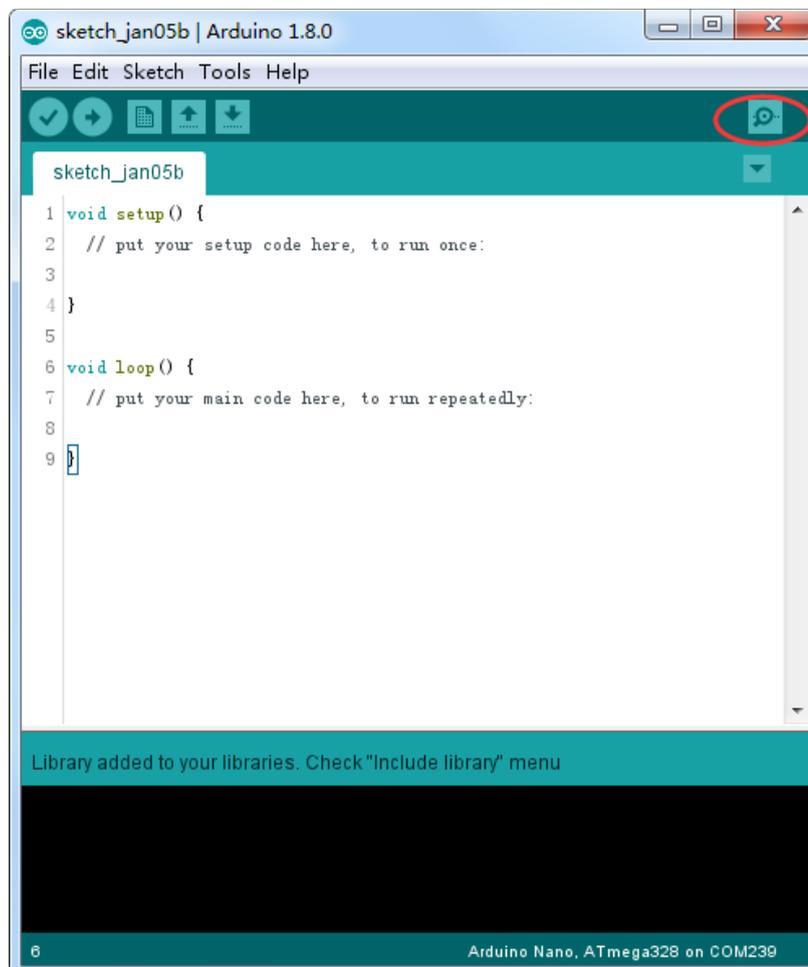
Wenn alles richtig ist, starten Sie die Arduino IDE neu. Stellen Sie sicher, dass die neue Bibliothek unter *Sketch > Bibliothek einbinden* zu finden ist. Dann sind Sie fertig.

Arduino Serieller Monitor (Windows, Mac, Linux)

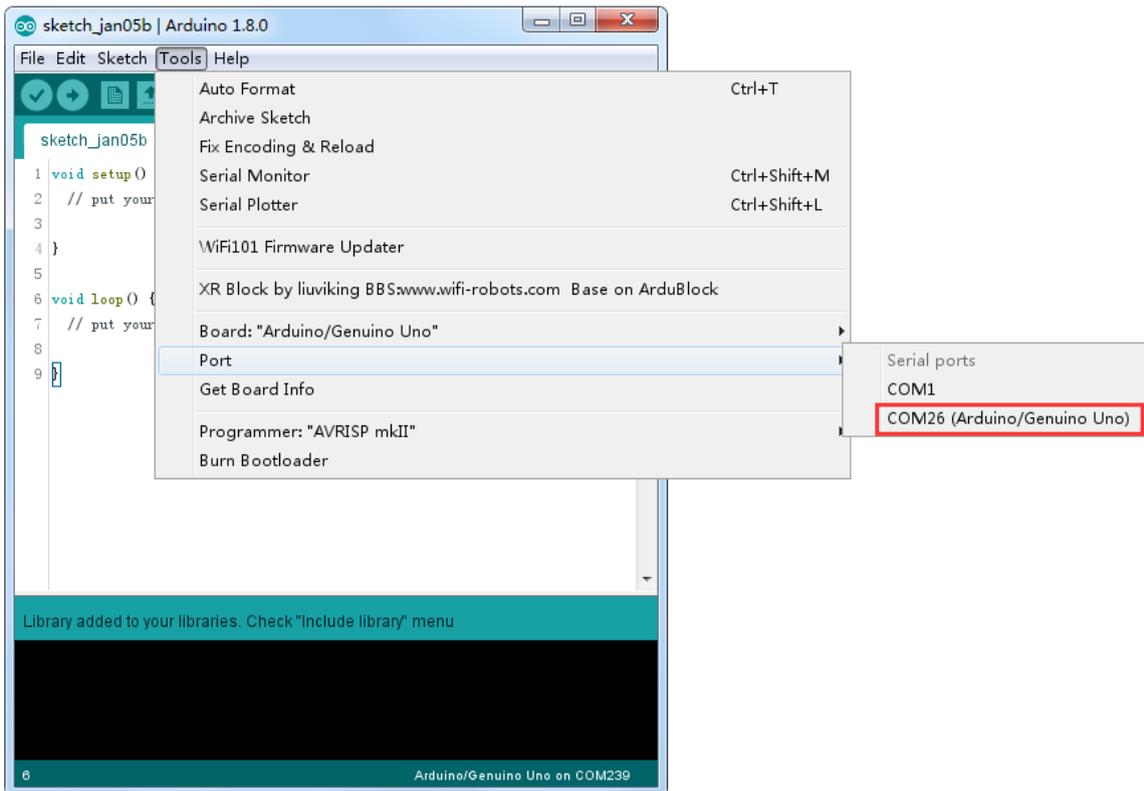
Die Arduino Entwicklungsumgebung (IDE) ist die Software Seite der Arduino Plattform. Und weil das Arbeiten mit einem Terminal ein so bedeutender Teil beim Programmieren mit Arduino und anderen Mikrocontrollern ist, haben sich die Entwickler der Arduino IDE dazu entschieden, ein Terminal (Kommandozeile) zu integrieren. In der Arduino Entwicklungsumgebung wird dies „Serieller Monitor“ genannt.

Die Verbindung herstellen

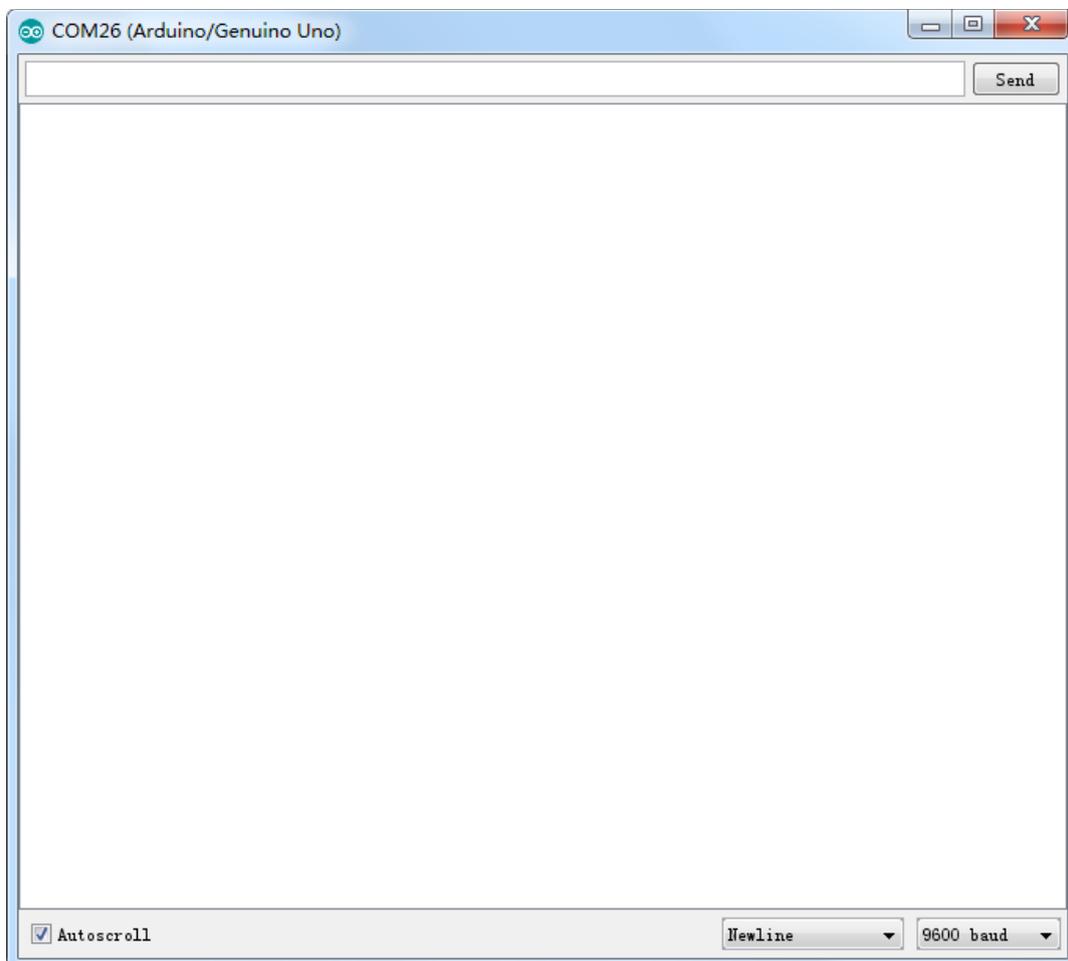
Der Serielle Monitor ist in jeder Version der Arduino IDE vorhanden. Um ihn zu öffnen, müssen Sie lediglich auf das im Bild unten markierte Symbol klicken.



Bevor die Verbindung möglich ist, müssen Sie den richtigen COM-Port auswählen. Gehen Sie zu *Werkzeuge* -> *Port* und wählen Sie den richtigen Port aus. Normalerweise ist nur einer vorhanden. Sonst schauen Sie im Geräte-Manager nach.

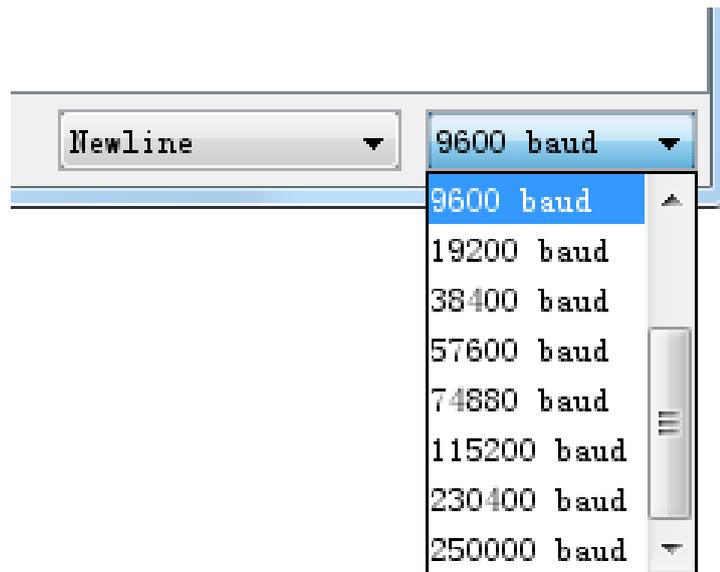


Sobald Sie den Seriellen Monitor dann geöffnet haben, erscheint dies:

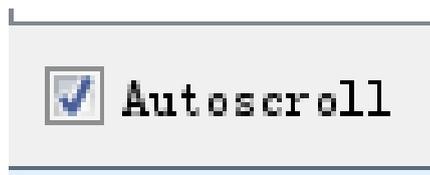


Einstellungen

Die Optionen des Seriellen Monitors sind sehr limitiert, aber reichen aus, um die wichtigsten Optionen der Seriellen Kommunikation festzulegen. Die erste Einstellung, die Sie machen, ist die Festlegung der *Baud-Rate*. Klicken Sie auf das im Bild zu sehene Drop-Down Menü und wählen Sie die richtige Baudrate aus (9600 baud).



Schließlich können Sie durch Anklicken der Checkbox *Autoscroll* einstellen, ob der Serielle Monitor automatisch mitscrollen soll.



Vorteile

Der Serielle Monitor ist ein schneller und einfacher Weg eine Serielle Verbindung zu Ihrem Arduino aufzubauen. Wenn Sie bereits mit der Arduino IDE arbeiten, brauchen Sie kein weiteres Terminal zu installieren, um sich die serielle Kommunikation Ihres Arduinos anzuschauen.

Nachteile

Die minimalistischen Einstellungen des Seriellen Monitors lassen zu wünschen übrig und für fortgeschrittene Serielle Kommunikation kann dies unzureichend sein.

Lektion 2: Blink

Übersicht

In dieser Lektion lernen Sie, wie Sie Ihr UNO R3 Entwicklungsboard so programmieren, dass die eingebaute LED leuchtet. Außerdem erfahren Sie Schritt für Schritt, wie man ein Programm auf das Arduino Board hochlädt.

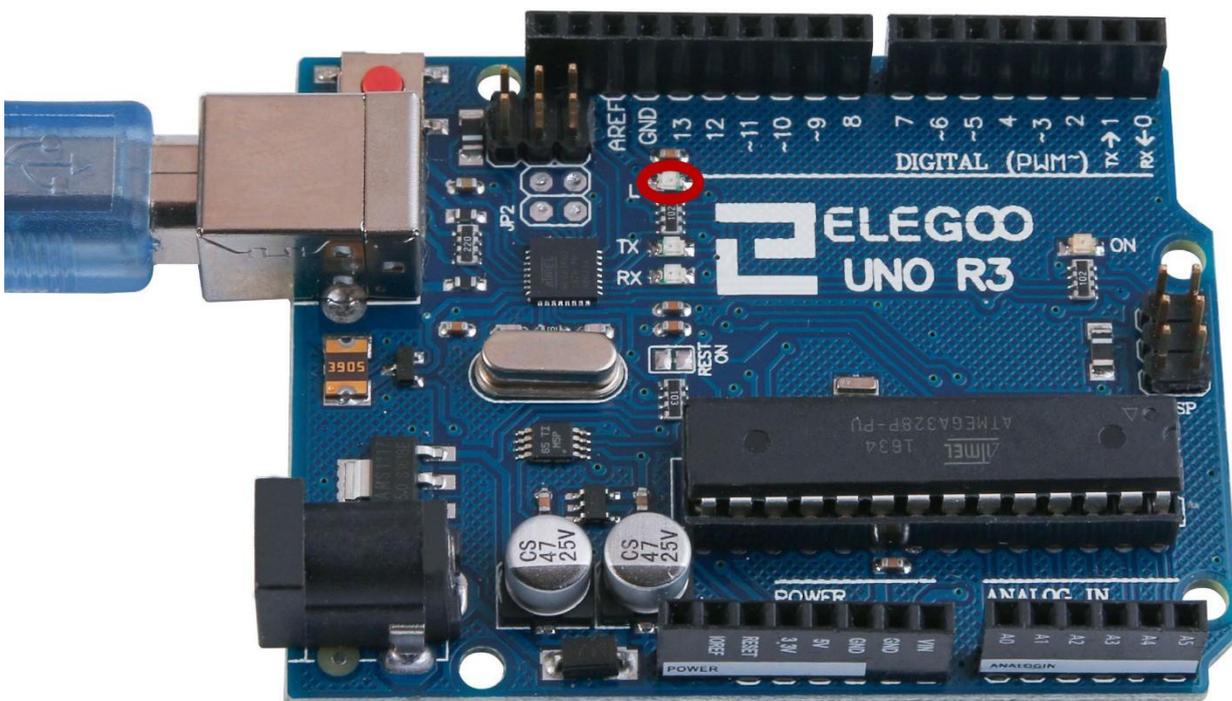
Benötigte Bauteile:

(1) x Elegoo UNO R3

Aufbau

Das UNO R3 Board hat an beiden Seiten Steckplätze, die dazu benutzt werden können, externe elektrische Geräte und Module („Shields“) anzuschließen, um so die technischen Möglichkeiten zu erweitern.

Das Board hat außerdem eine LED, die Sie in Ihren Sketches (Programmen) kontrollieren können. Diese LED ist fest im UNO R3 Board verbaut und wird oft „L“-LED genannt, da sie so auf dem Board gekennzeichnet ist.



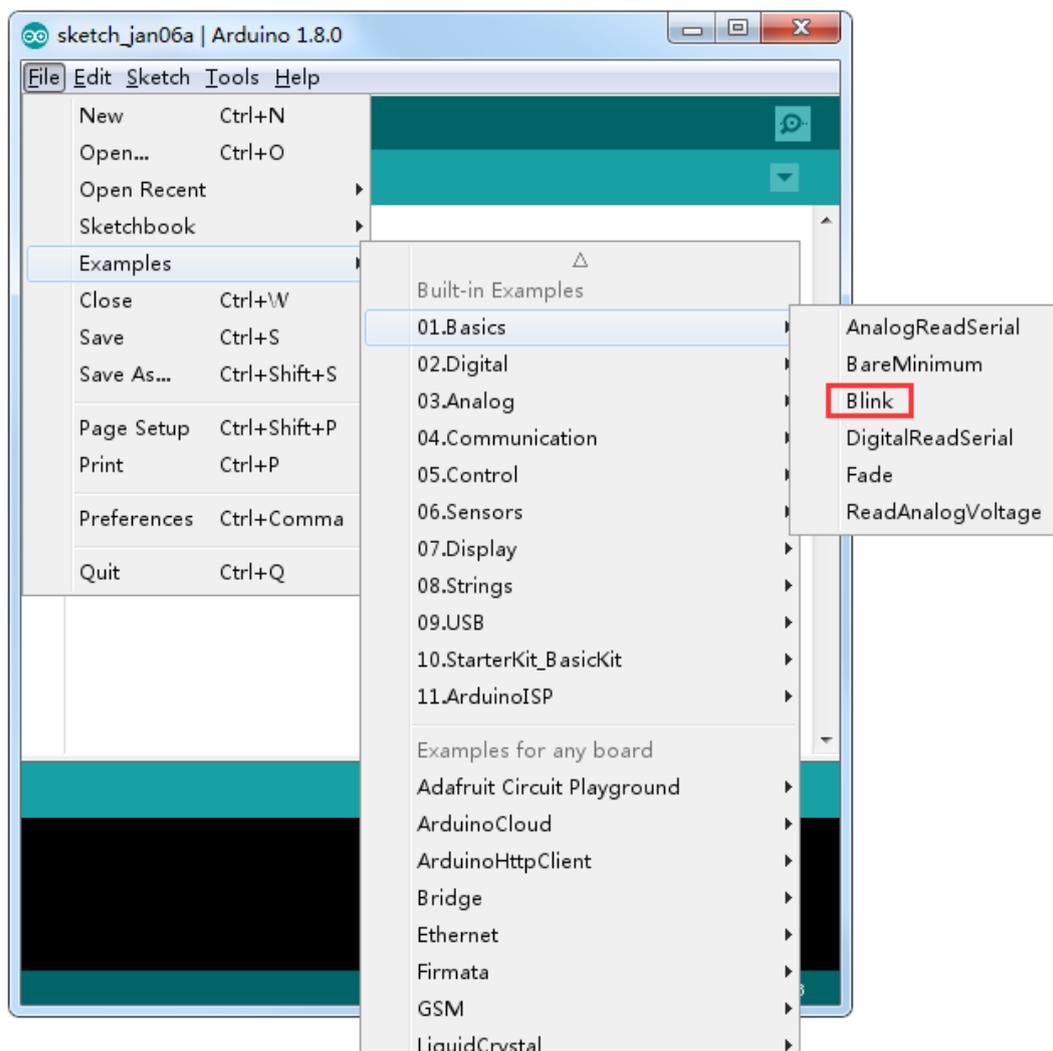
Es kann sein, dass Ihre „L“-LED bereits blinkt, wenn Sie Ihr UNO R3 Board mit einer Stromversorgung verbinden. Das liegt daran, dass die ausgelieferten Boards bereits mit dem „Blink“-Sketch vorinstalliert kommen, der die LED blinken lässt.

In dieser Lektion werden wir das UNO R3 Board mit unserem eigens kreierte Blink Sketch programmieren und anschließend die Blinkgeschwindigkeit ändern.

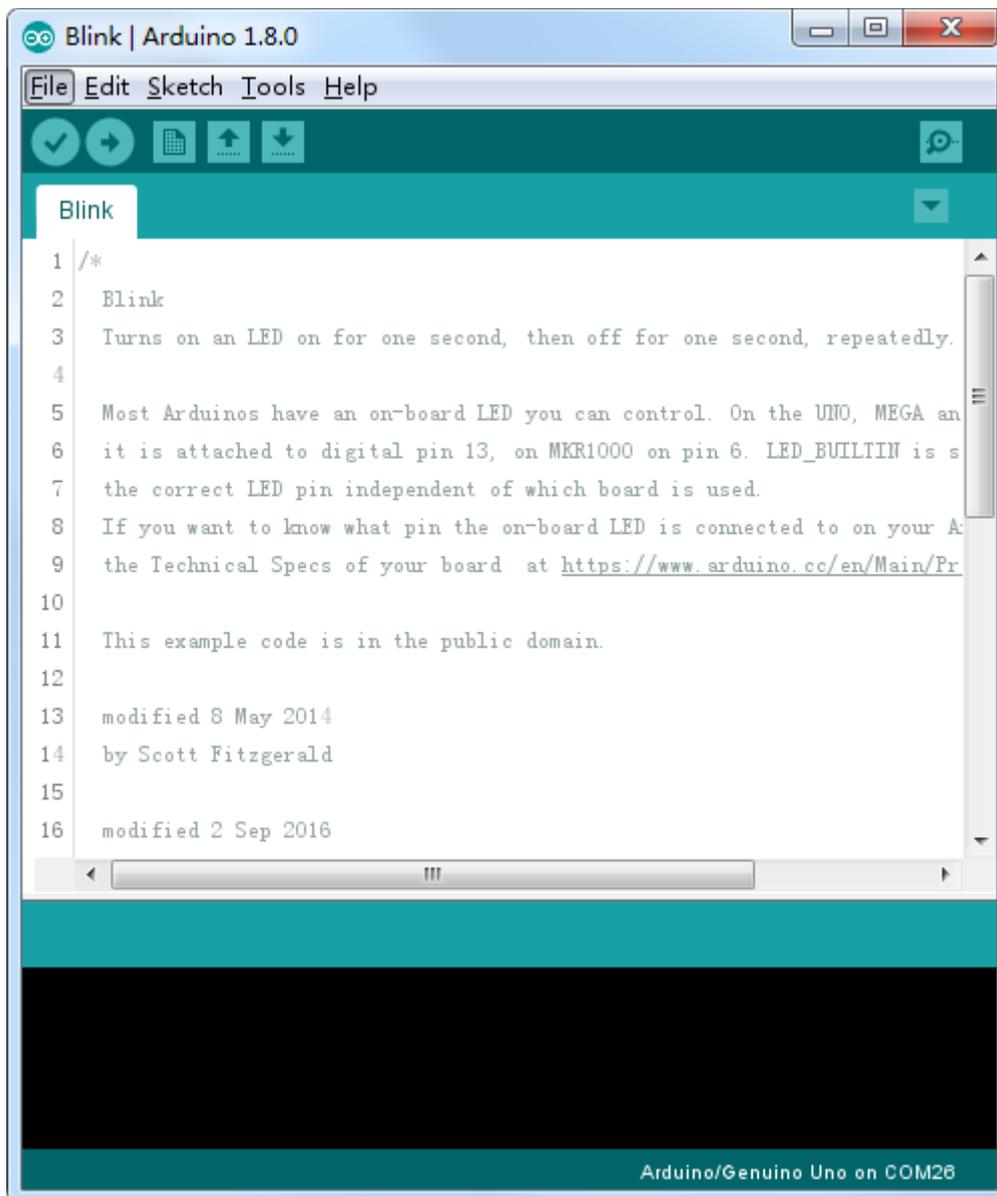
In Lektion 0 haben Sie die Arduino IDE (Entwicklungsumgebung) aufgesetzt und den richtigen Seriellen Port in der IDE eingestellt. Nun werden wir diese Verbindung testen, indem wir das UNO R3 Board erstmals selber programmieren.

Die Arduino IDE kommt mit einer großen Sammlung von Beispiel-Sketches, die Sie einfach öffnen und auf das Board hochladen können. In dieser Sammlung befindet sich auch bereits ein Sketch, der die L-LED zum leuchten bringt.

Laden Sie den Blink-Sketch, den Sie in der IDE unter *Datei > Beispiele > 0.1Basics* finden.



Wenn sich der Blink-Sketch geöffnet hat, sollten Sie das Fenster vergrößern, um sich einen Überblick über den gesamten Programm-Code schaffen zu können.



The screenshot shows the Arduino IDE interface with the 'Blink' sketch open. The window title is 'Blink | Arduino 1.8.0'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for saving, running, uploading, and downloading. The sketch editor displays the following code:

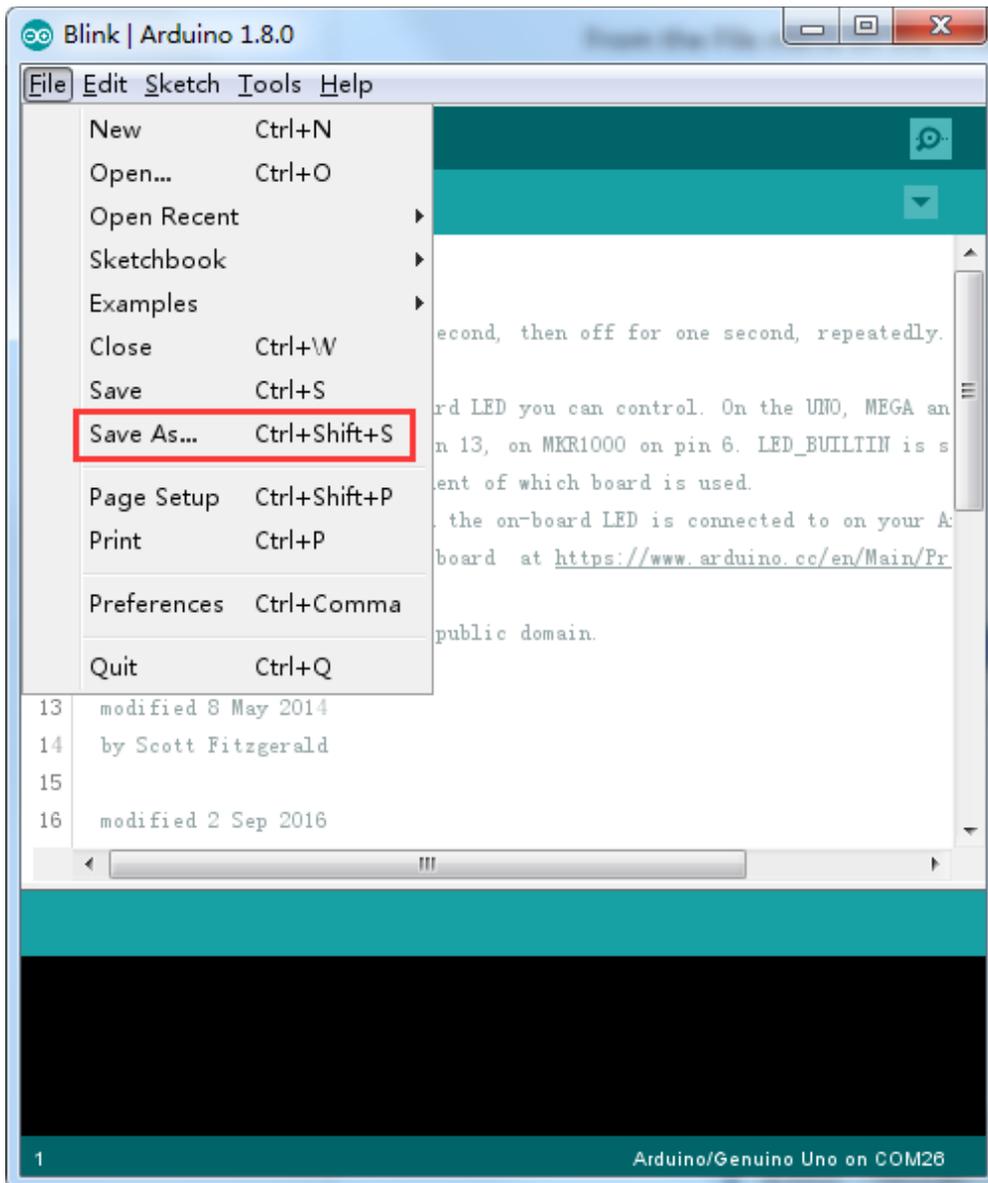
```
1 /*
2  Blink
3  Turns on an LED on for one second, then off for one second, repeatedly.
4
5  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and
6  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is s
7  the correct LED pin independent of which board is used.
8  If you want to know what pin the on-board LED is connected to on your A
9  the Technical Specs of your board at https://www.arduino.cc/en/Main/Pr
10
11  This example code is in the public domain.
12
13  modified 8 May 2014
14  by Scott Fitzgerald
15
16  modified 2 Sep 2016
```

The status bar at the bottom right indicates 'Arduino/Genuino Uno on COM26'.

Die mitinstallierten Beispiel-Sketches sind nicht „schreibbar“. Das heißt, man kann Änderungen nicht abspeichern. Sie können den Sketch abändern und hochladen, aber nicht unter dem gleichen Dateinamen speichern.

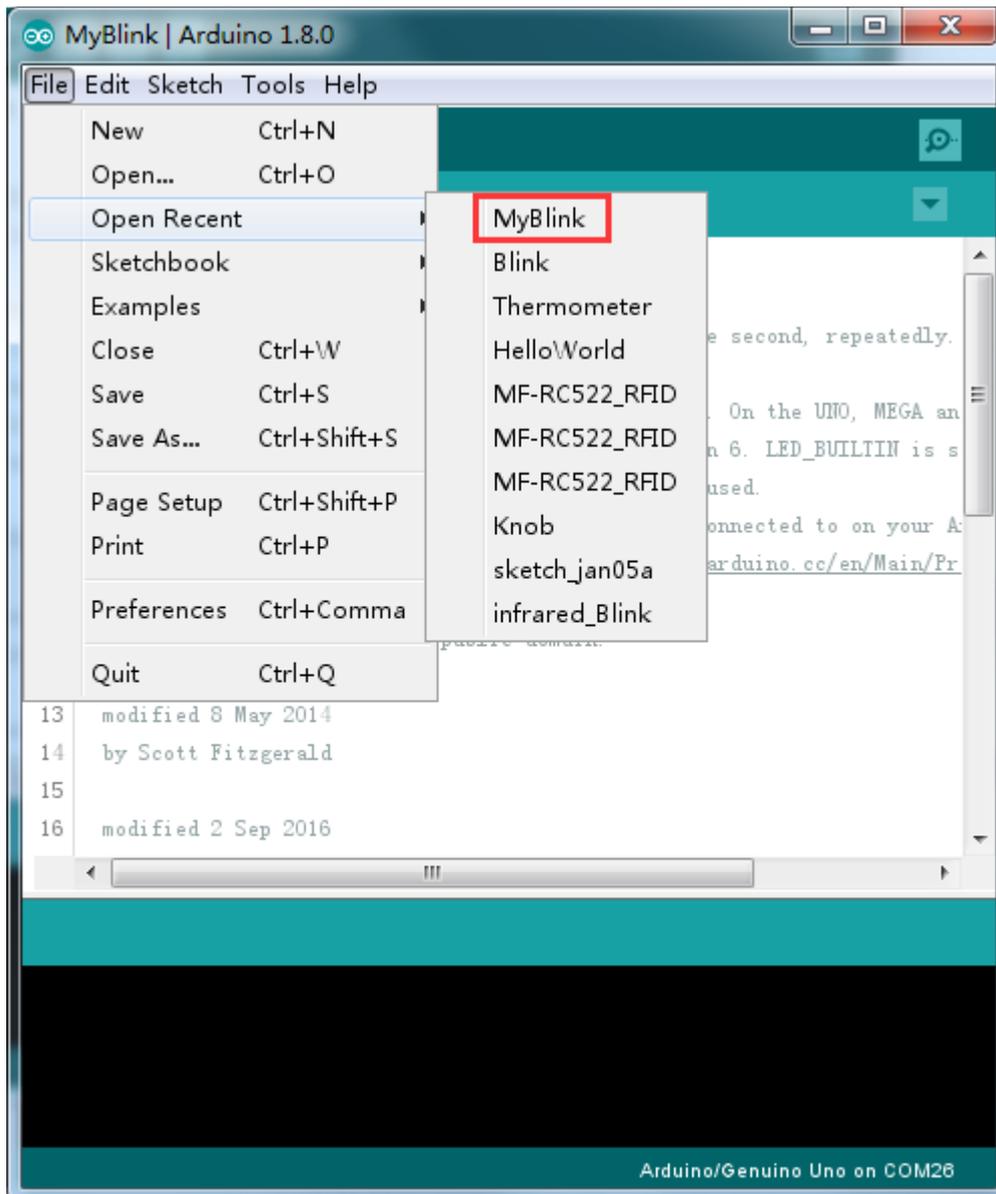
Da wir den Sketch ändern möchten, ist der erste Schritt den Sketch unter neuem Namen abzuspeichern.

Im Arduino IDE Menü wählen Sie *Datei > Speichern unter...* aus und speichern den Sketch dann unter dem Namen „MyBlink“.

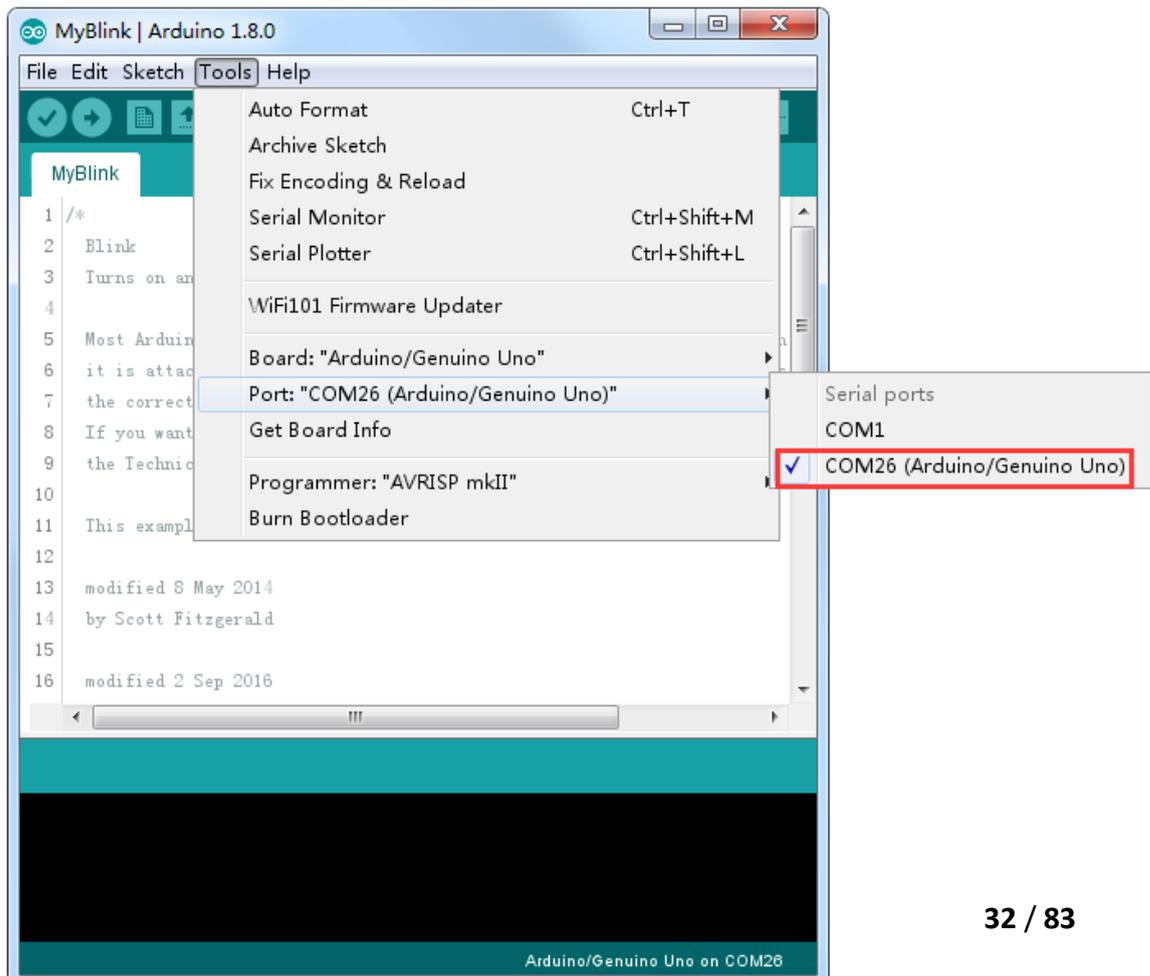
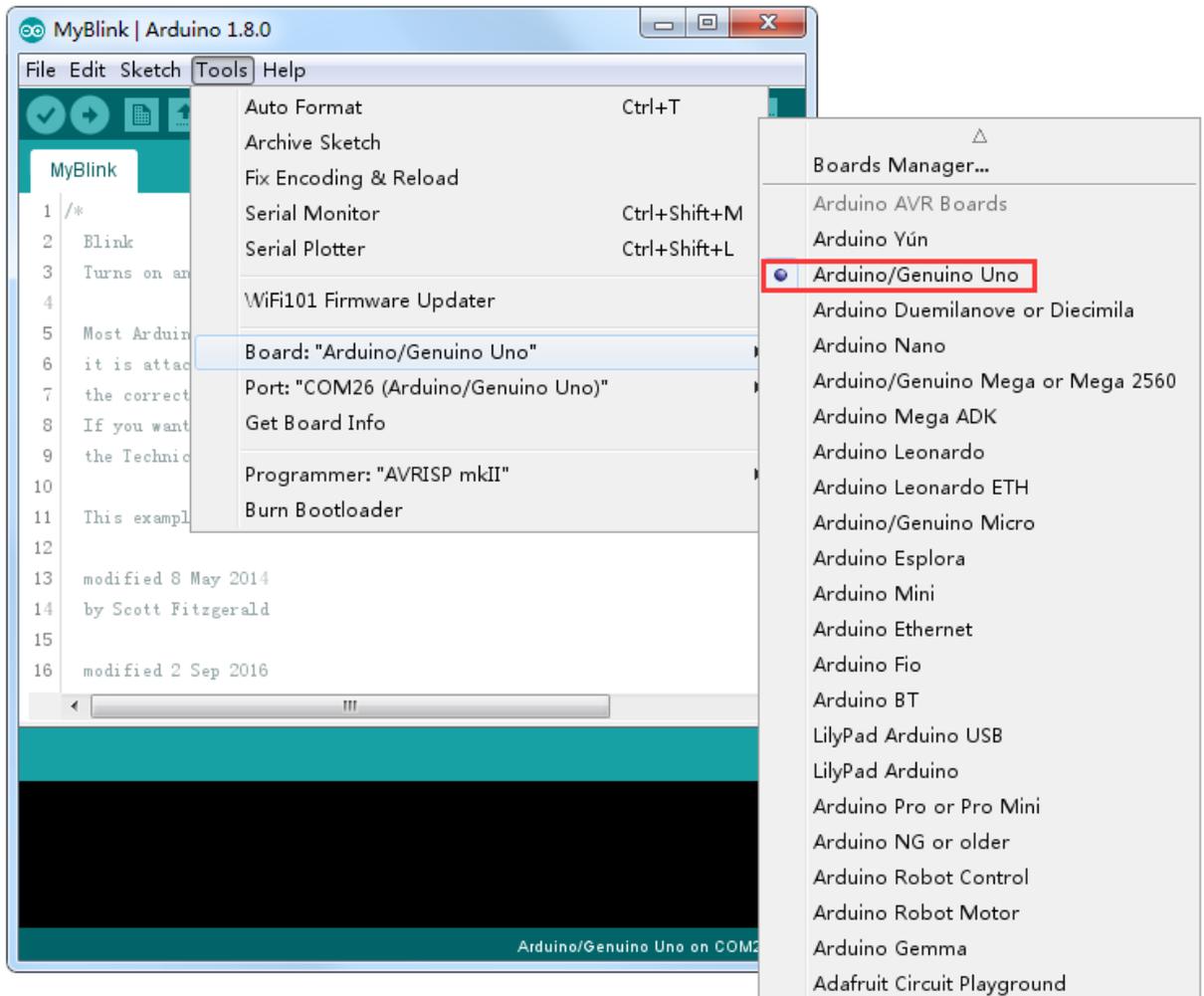




Sie speichern Ihre Kopie des Blink-Sketches in Ihrem Sketchbook (Projektordner) ab. Das bedeutet, dass Sie bei Bedarf den Sketch einfach über *Datei > Sketchbook* im Menü aufrufen können.

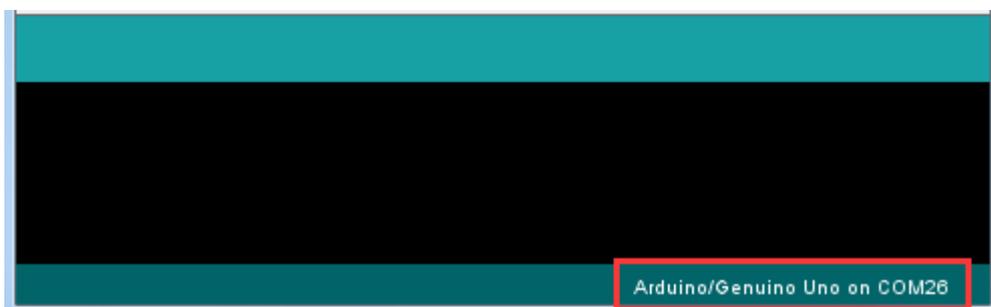


Verbinden Sie Ihr Arduino Board mit einem USB-Kabel mit dem Computer und überprüfen Sie, ob der „Board Typ“ und der „Serielle Port“ richtig eingestellt sind,



Achtung: Ihr Board-Typ und der Serielle Port kann sich von den in den Bildern zu sehenden Angaben unterscheiden. Wenn Sie das 2560-Board verwenden, dann wählen Sie „Mega 2560“ als Board-Typ, wenn Sie das UNO R3 Board verwenden, wählen Sie „Uno“ als Board-Typ, usw. Der Serielle Port ist bei jedem Gerät unterschiedlich und wird vom Computer zugewiesen. Im Gegensatz zum hier angegebenen Port COM26, könnte es bei Ihnen COM3 oder COM4 sein. Der richtige COM-Port ist durch ein „COMX (arduino XXX)“ gekennzeichnet.

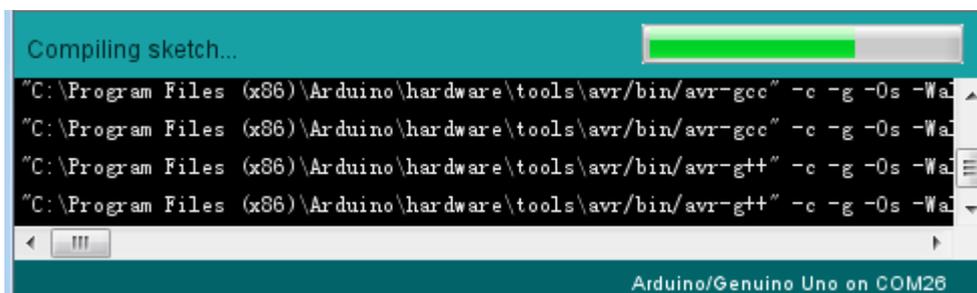
Die Arduino IDE zeigt die aktuellen Einstellungen für das Board im Fenster rechts unten an.



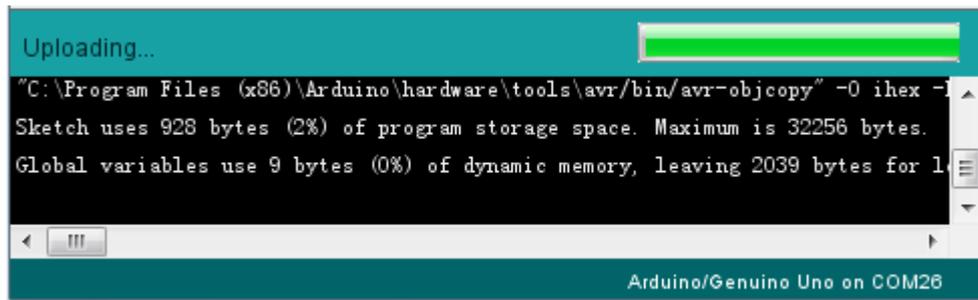
Klicken Sie nun auf die „Hochladen“-Schaltfläche. Dies ist das zweite Symbol von links.



Wenn Sie den Statusbereich der IDE in der unteren Sektion des Fensters beobachten, sehen Sie, wie sich eine Fortschrittsleiste langsam füllt und Meldungen erscheinen. Zuerst erscheint die Meldung „*Sketch wird kompiliert...*“. Dabei wird der Sketch in ein für das Board passende Format umgewandelt.



Als nächstes ändert sich der Status zu „*Hochladen*“. An diesem Punkt sollten die LEDs des UNO R3 Boards anfangen zu blinken, da der Sketch auf das Board geladen wird.



Zum Schluss ändert sich der Status zu „Hochladen abgeschlossen“.



Die Meldung darunter teilt uns mit, dass der Sketch 928 Bytes von 32.256 Bytes (der gesamte verfügbare Speicher) belegt. Es ist möglich, dass Sie stattdessen nach dem Schritt „Sketch wird kompiliert...“ eine orange Fehlermeldung erhalten:



Das kann zum einen bedeuten, dass Ihr Board gar nicht mit dem Computer verbunden ist oder aber die Treiber wurden (bei manueller Installation) nicht mitinstalliert. Zuletzt kann auch ein falsch ausgewählter Serieller Port für die Fehlermeldung verantwortlich sein.

Wenn dieses Problem bei Ihnen auftritt, gehen Sie zurück zu Lektion 0 und überprüfen Sie Ihre Arduino IDE Installation.

Hat dagegen alles geklappt, sollte das Board nach Abschluss neustarten und anfangen zu blinken. Öffnen Sie den Sketch.

Beachten Sie, dass ein großer Teil dieses Sketches aus Kommentaren besteht. Dabei handelt es sich um keinen richtigen Code, also Anweisungen, was das Board zu tun

hat, sondern nur um Kommentare, die dem Programmierer erklären, wie der Sketch funktioniert. Die Kommentare dienen Ihnen zur besseren Verständlichkeit.

Alles zwischen den Zeichen `/*` und `*/` am Anfang des Sketches ist ein Block-Kommentar. Dort wird erklärt, wozu der Sketch dient.

Ein-Zeilen-Kommentare starten mit `//` und alles von diesem Zeichen an bis hin zum Ende der Zeile wird als Kommentar interpretiert.

Die erste Zeile des Codes lautet:

```
int led = 13;
```

Wie der Kommentar darüber erklärt, weist dies dem Pin der LED einen Namen zu (*led*). Dieser Pin ist der Pin 13 auf den meisten Arduinos, wie beispielsweise UNO und Leonardo.

Als nächstes haben wir die „*setup*“-Funktion. Der Kommentar erklärt, dass alles was sich in dieser Funktion befindet ausgeführt wird, wenn der *Reset*-Knopf gedrückt wird. Es wird außerdem ausgeführt, wenn das Board aus irgendeinem anderen Grund resettet wird, wie zum Beispiel nach dem Einstecken der Stromversorgung oder nach dem Hochladen eines Sketches.

```
void setup() {  
  // initialize the digital pin as an output.  
  pinMode(led, OUTPUT);  
}
```

Jeder Arduino Sketch muss eine *setup*-Funktion haben. Der Code für diese Funktion muss zwischen den geschwungenen Klammern `{ }` platziert werden.

In diesem Fall gibt es in der *setup*-Funktion nur einen einzigen Befehl, welcher bestimmt, dass das Arduino Board den LED-Pin als einen Ausgang benutzen soll.

Außerdem grundlegend für jeden Sketch ist die „*loop*“-Funktion. Im Gegensatz zur *setup*-Funktion, die nur einmalig nach einem Reset läuft, wiederholt sich die *loop*-Funktion ständig, nachdem Sie alle ihre Befehle ausgeführt hat (*loop* = Schleife).

```
void loop() {  
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);             // wait for a second  
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);             // wait for a second  
}
```

Innerhalb der loop-Funktion wird durch die Befehle der LED-Pin erstmal angeschaltet (HIGH). Darauf folgt eine 1000 Milisekunden (= 1 Sekunde) lange Pause. Dann wird der LED-Pin wieder ausgeschaltet (LOW) und es findet wieder eine 1-sekündige Pause statt.

Jetzt werden Sie die LED dazu bringen schneller zu blinken. Wie Sie bereits geahnt haben könnten, liegt der Schlüssel der Blinkgeschwindigkeit darin, die Zeiten der Pausen anzupassen (die Zahlen innerhalb der Klammern der „*delay*“-Befehle).

```
30 // the loop function runs over and over again forever
31 void loop() {
32   digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the volt
33   delay(500) // wait for a second
34   digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the vo
35   delay(500) // wait for a second
36 }
```

Die Zeitspanne der Pausen wird in Milisekunden angegeben (1s = 1000ms). Wenn Sie also die LED doppelt so schnell blinken lassen wollen, müssen Sie die Werte halbieren, also von 1000 auf 500 ms senken. Das sorgt dafür, dass zwischen Ein- und Ausschalten der LED nur noch jeweils eine halbe Sekunde Pause stattfindet.

Laden Sie den Sketch erneut hoch und am Ende sollten Sie die LED schneller blinken sehen.

Sie können die Werte nun beliebig anpassen und den Sketch erneut hochladen und beobachten, wie sich die Blinkgeschwindigkeit jedes mal ändert.

Lektion 3: LEDs

Übersicht

In dieser Lektion werden Sie lernen, wie man die Helligkeit von LEDs mit Hilfe verschiedener Widerstände verändern kann.

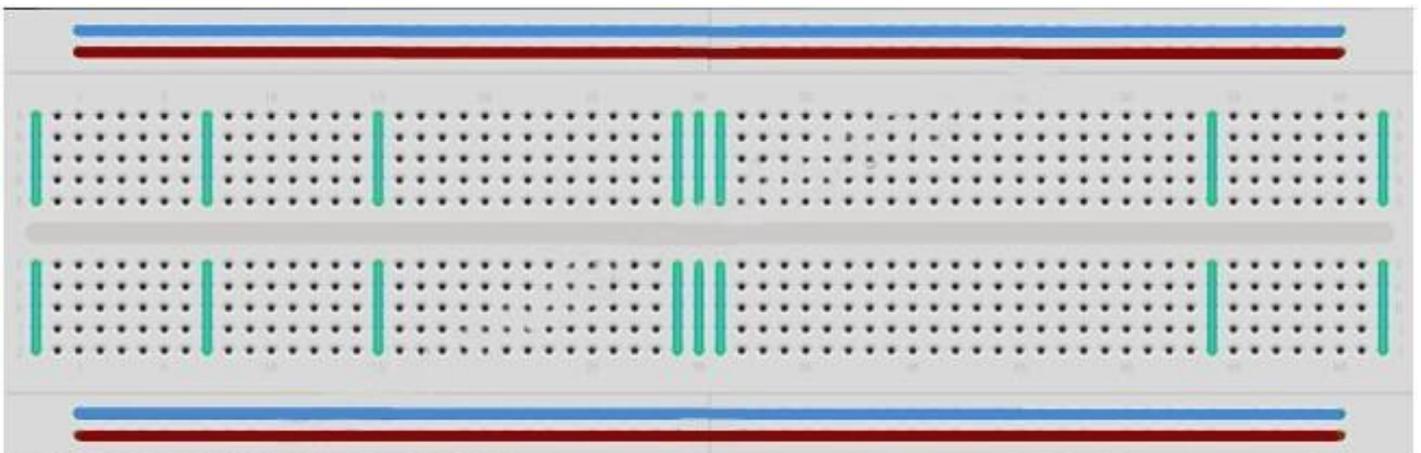
Benötigte Bauteile:

- (1) x Elegoo UNO R3
- (1) x 5mm rote LED
- (1) x 220 Ohm Widerstand
- (1) x 1k Ohm Widerstand
- (1) x 10k Ohm Widerstand
- (2) x M-M Kabel (Männlich zu Männlich DuPont Jumper Kabel)

Einführung in die Komponenten

BREADBOARD MB-102:

Ein Breadboard (Steckplatine) ermöglicht es Ihnen, schnell und einfach Schaltungen zu testen, ohne dass gelötet werden muss. Unten finden Sie ein Beispielmodell.



Breadboards gibt es in zahlreichen Größen und Modellen. Die einfachste Art ist ein simples Netz aus Löchern in einem Plastikblock. Innerhalb des Blocks befinden sich Metallstreifen, die die elektrische Verbindung zwischen den Kontaktstellen herstellen. Dabei sind immer die Kontakte innerhalb der selben Reihe (die kurze Seite entlang) miteinander verbunden. Wenn man die Pins von zwei verschiedenen Bauteilen also in die selbe Kontaktreihe steckt, sind sie elektrisch miteinander verbunden. In der Mitte des Blocks befindet sich längs eine Unterbrechung der Kontakte, sodass man die Kontaktreihen links und rechts der Unterbrechung unabhängig voneinander nutzen kann. Manche Breadboards haben an den beiden Außenseiten zwei gesonderte Kontaktreihen, die senkrecht zu den restlichen verlaufen. Diese länglichen Kontaktreihen nennt man „*Rails*“ (Schienen) und sind dazu da, um die verwendeten Bauteile mit Spannung (meist 5 Volt) zu versorgen. Während Breadboards sich gut zum Austesten von Prototypen eignen, haben sie einige Limitierungen. Weil die Verbindungen auf Druck bzw eingeklemmten Kabeln basieren und nur temporär sind, sind sie nicht so zuverlässig und haltbar wie gelötete Verbindungen. Wenn bei Ihnen unregelmäßig Fehler in einem Schaltkreis auftreten, könnte es an einem Wackelkontakt auf dem Breadboard liegen.

LEDs:

LEDs (Licht Emittierende Dioden) lassen sich gut als Indikatoren benutzen. Sie verbrauchen sehr wenig Strom und halten so gut wie ewig.

In dieser Lektion werden Sie die wahrscheinlich häufigst vorkommende aller LEDs benutzen: eine 5mm große rote LED. 5mm meinen dabei den Durchmesser der LED. Andere häufige Größen sind 3mm und 10mm.

Man kann eine LED nicht irgendwie an eine Batterie oder Spannungsquelle anschließen. LEDs haben einen positiven und einen negativen Anschluss und leuchten nur auf, wenn sie richtig verbunden sind. Außerdem müssen LEDs mit einem Widerstand benutzt werden, um den fließenden Strom zu begrenzen.



Ohne vorgeschalteten Widerstand wird die LED zerstört, da dadurch zu viel Strom durch sie fließt. Zu hoher Strom erhitzt die LED und lässt sie durchbrennen.

Es gibt zwei Wege zu erkennen, welches der positive und welches der negative Anschluss ist.

Das erste ist die Länge: Der positive Anschluss ist immer länger als der negative.

Außerdem ist an der Stelle, wo der negative Anschluss in das Gehäuse der LED führt, ein abgeflachter Kontakt.

Wenn Sie also eine LED haben, dessen abgeflachter Kontakt gegenüber dem langen Anschluss liegt, ist der lange Anschluss der positive.

Widerstände:

Wie der Name vermuten lässt, setzen Widerstände dem fließenden Strom einen Widerstand entgegen. Je höher der Wert eines Widerstandes, desto mehr Strom wird „aufgehalten“ und desto weniger elektrischer Strom fließt durch die Leitung. Wir benutzen hier einen Widerstand, um zu beeinflussen, wie viel Strom durch die LED fließen wird und daher auch wie hell sie leuchten wird.

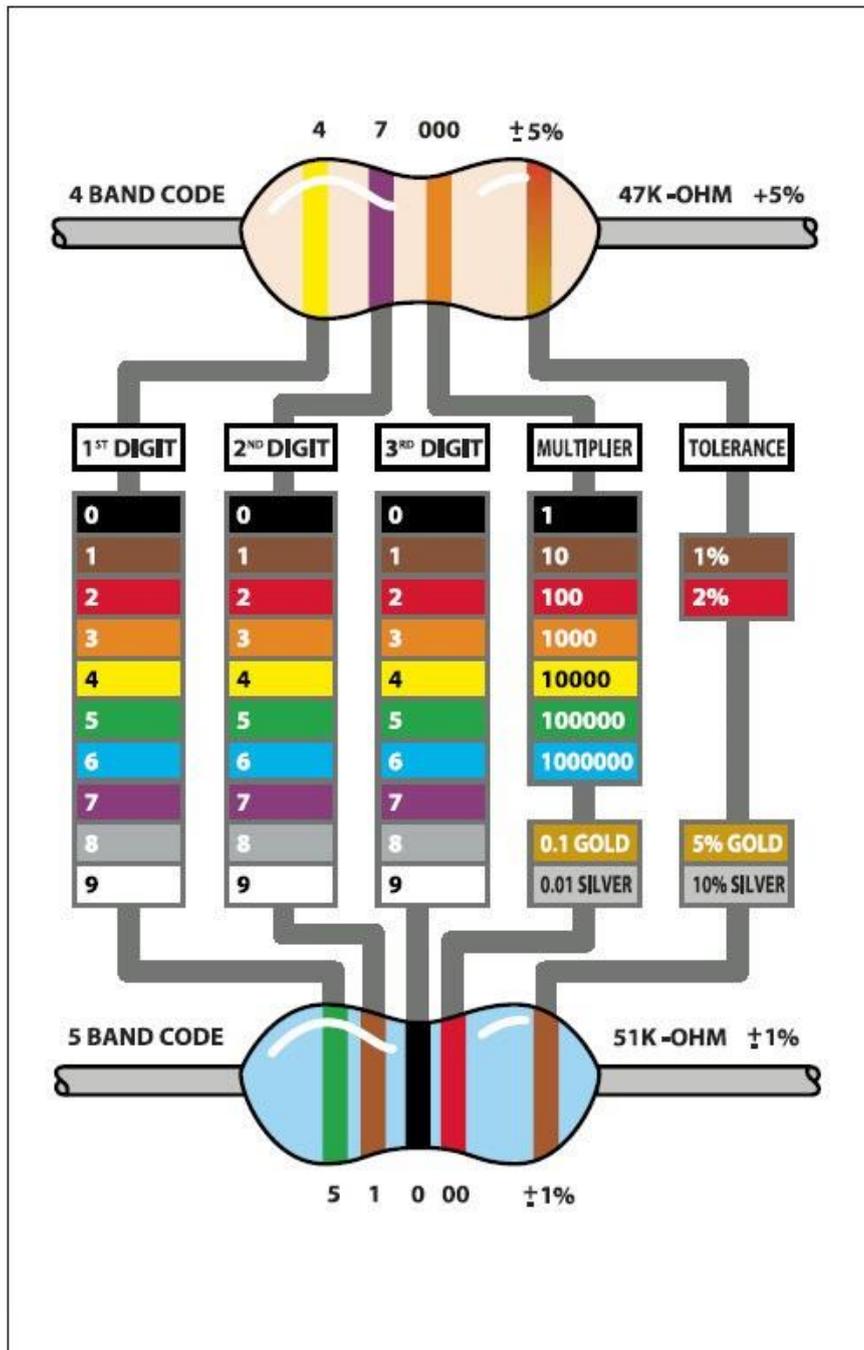


Aber zuerst mehr über Widerstände:

Die Einheit von Widerständen wird „*Ohm*“ genannt, was mit dem griechischen Buchstaben Ω Omega abgekürzt wird. Da 1 Ohm einem sehr geringen Widerstand entspricht, gibt es auch Widerstände in Größen von $k\Omega$ (1.000 Ω) und $M\Omega$ (1.000.000 Ω). Diese Größen werden Kilo-Ohm und Mega-Ohm genannt.

In dieser Lektion benutzen wir drei verschiedene Widerstände: 220 Ω , 1k Ω und 10k Ω . Diese Widerstände sehen, bis auf den aufgemalten Farbcode, alle ziemlich gleich aus. Die bunten Streifen auf dem Widerstand repräsentieren den Wert des Widerstands und könne in diesen mit Hilfe einer Tabelle umgerechnet werden.

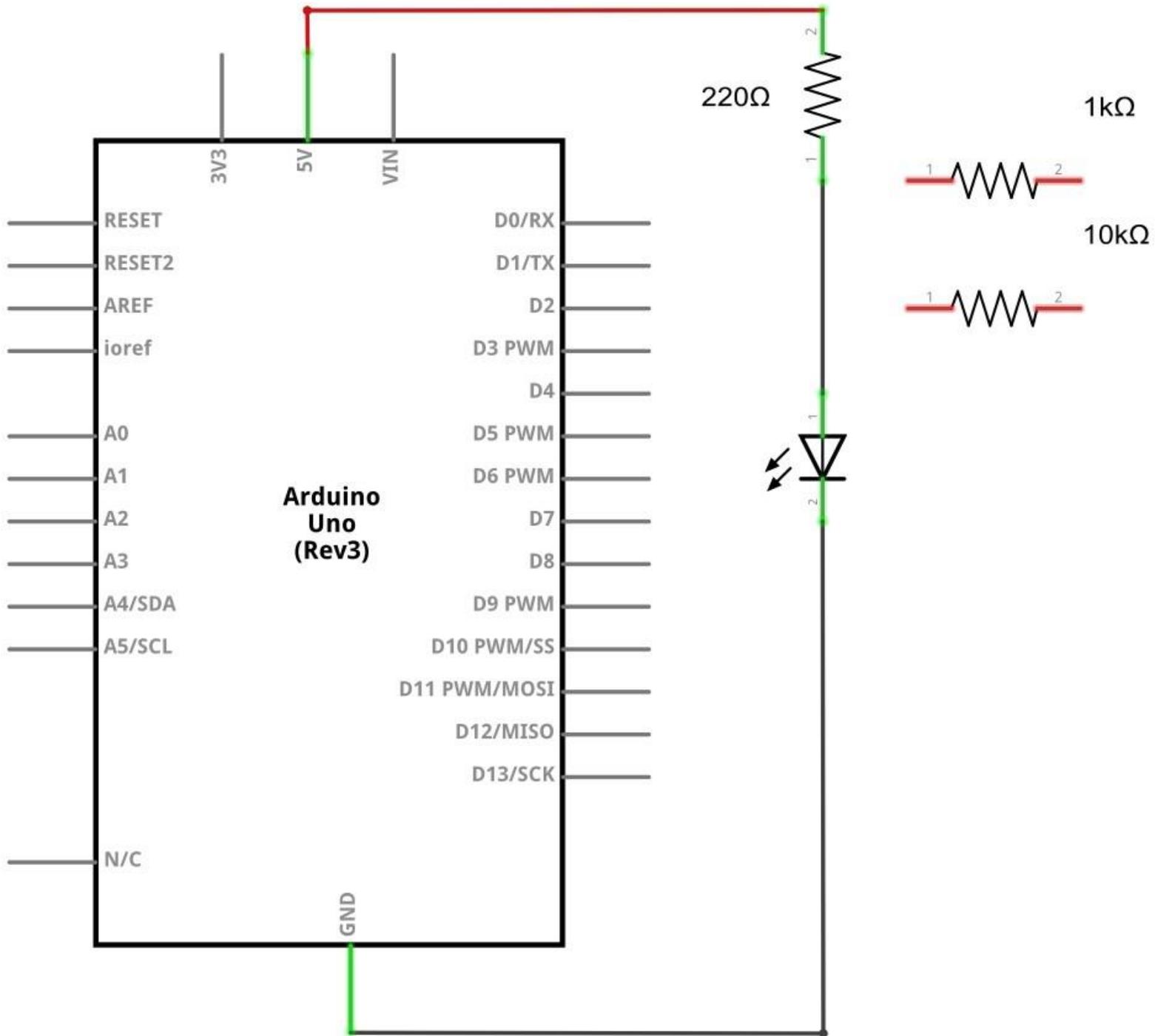
Der Farbcode unseres Widerstandes hat drei farbige Streifen und einen goldenen am Ende.



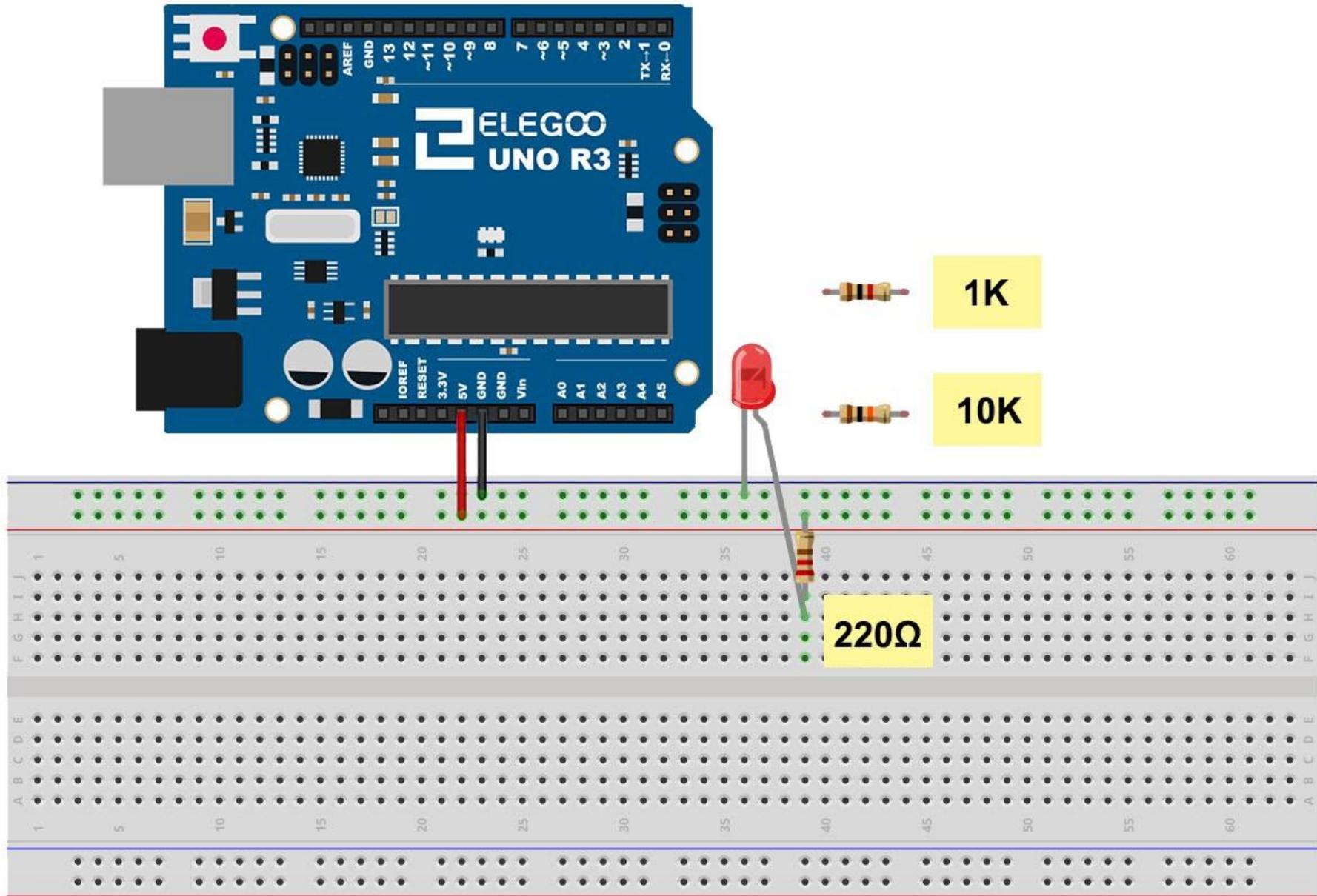
Im Gegensatz zu LEDs haben Widerstände keine positiven und negativen Enden und können daher egal in welche Richtung verbaut werden.

Wenn Ihnen die Methode mit der Tabelle zu kompliziert erscheint, können Sie den Farbcode auf unseren Widerständen direkt ablesen, um seinen Wert zu bestimmen. Alternativ können Sie einen Widerstand auch mit einem Multimeter durchmessen und so den Wert bestimmen.

Verbindungsschema



Schaltplan



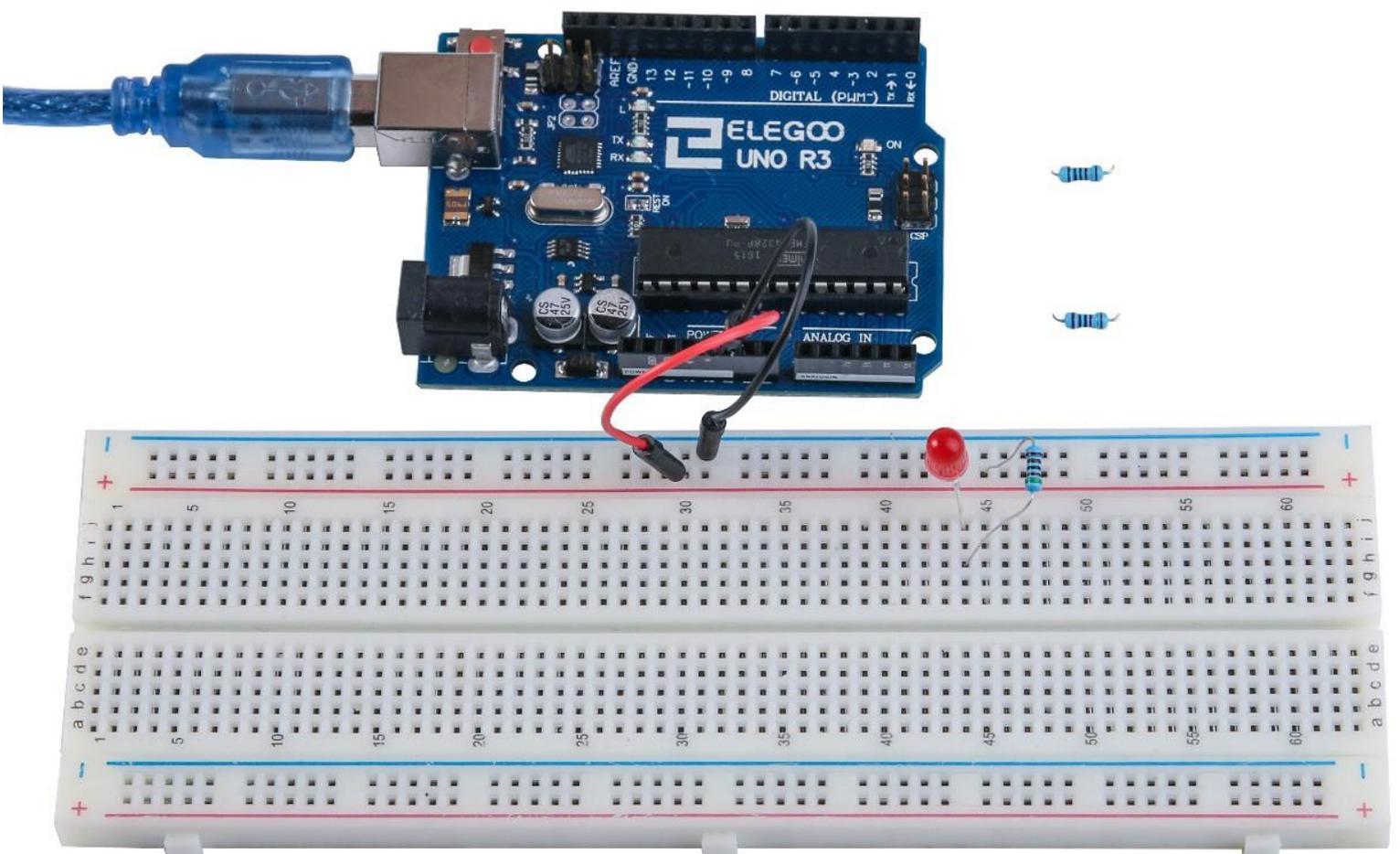
Das UNO Board ist eine praktische Quelle, um 5 Volt Spannung zu erhalten, die wir dafür benutzen werden, um die LED und den Widerstand mit Strom zu versorgen. Sie müssen mit Ihrem Board nichts weiter tun, als es mit einem USB-Kabel zu verbinden.

Mit dem vorgeschalteten 220Ω Widerstand sollte die LED recht hell leuchten. Wenn Sie den 220Ω Widerstand durch den $1k\Omega$ Widerstand ersetzen, sollte die LED ein bisschen dunkler erscheinen. Schließlich, mit dem vorgeschalteten $10k\Omega$ Widerstand, wirkt die LED fast komplett dunkel. Stecken Sie zeitweise das rote Jumper Kabel aus dem Breadboard und wieder ein, um den Unterschied zu einer stromlosen LED zu sehen.

Im Moment haben Sie die positive 5V Spannung mit einem Ende des Widerstandes verbunden, dessen anderes Ende mit der positiven Seite der LED verbunden ist. Die negative Seite der LED geht schließlich an den negativen Anschluss der Stromquelle (auch *GND* = Ground genannt). Wie auch immer, wir können den Widerstand auch so verschieben, dass er nach der LED kommt und es wird trotzdem funktionieren.

Sie können nun den 220Ω Widerstand wiedereinsetzen. Es ist egal an welcher Stelle der LED wir den Widerstand setzen, solange er mit ihr in Reihe geschaltet ist.

Beispielbild



Lektion 4: RGB LEDs

Übersicht

RGB LEDs sind ein einfacher Weg, um Farbe in Ihre Projekte zu bringen. Da sich in einer RGB-LED nichts anderes als drei reguläre LEDs in einem befinden, ist die Handhabung dieser ähnlich.

Es gibt sie in zwei Versionen: Mit gemeinsamer Anode und mit gemeinsamer Kathode. Bei gemeinsamer Anode liegt an dem gemeinsam genutzten Pin 5V an, bei gemeinsamer Kathode wird der gemeinsame Pin mit GND (Ground bzw. Erdung) verbunden.

Wie bei jeder LED müssen einige Widerstände vorgeschaltet werden (3 insgesamt), damit der fließende Strom begrenzt wird.

In unserem Sketch lassen wir die LED zuerst in rot leuchten, lassen sie dann nach grün übergehen, danach nach blau und schließlich zurück zur roten Farbe. Dabei werden wir durch den Übergang zwischen den Farben durch einen Großteil aller möglichen Farben schalten.

Benötigte Bauteile:

- (1) x Elegoo UNO R3
- (1) x 830 Punkte Breadboard
- (4) x M-M Kabel (Männlich zu Männlich DuPont Jumper Kabel)
- (1) x RGB LED
- (3) x 220 Ohm Widerstände

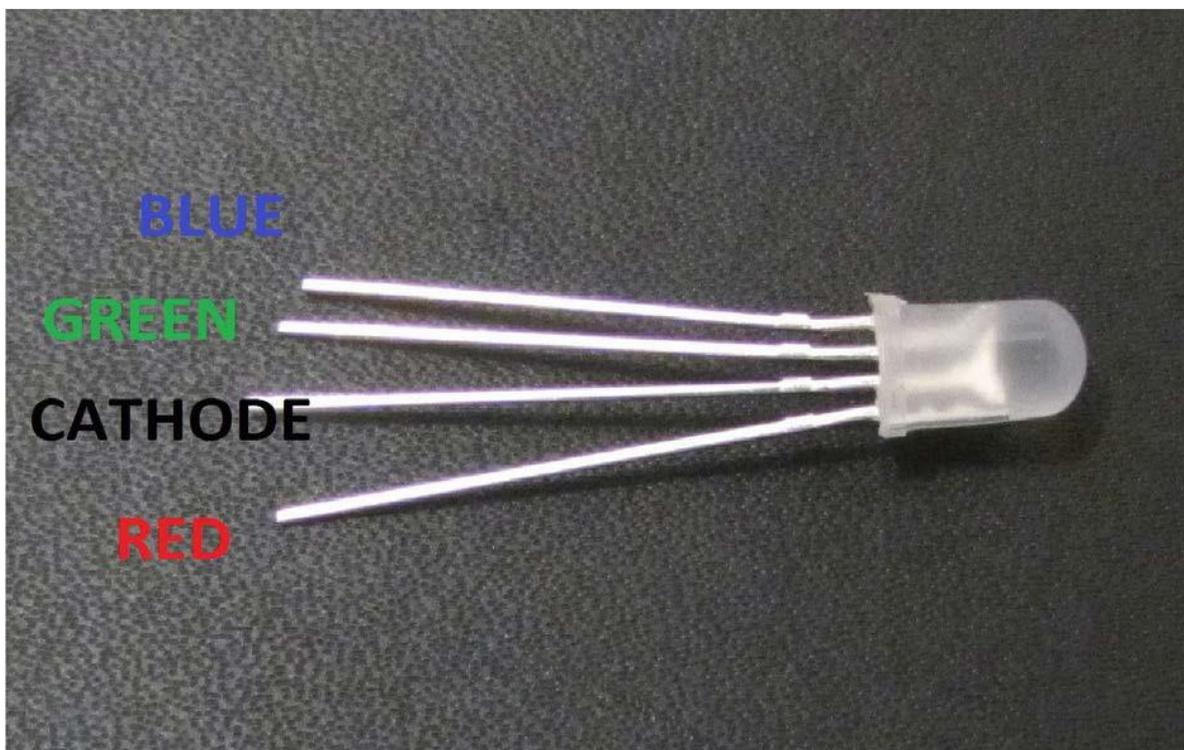
Einführung in die Komponenten

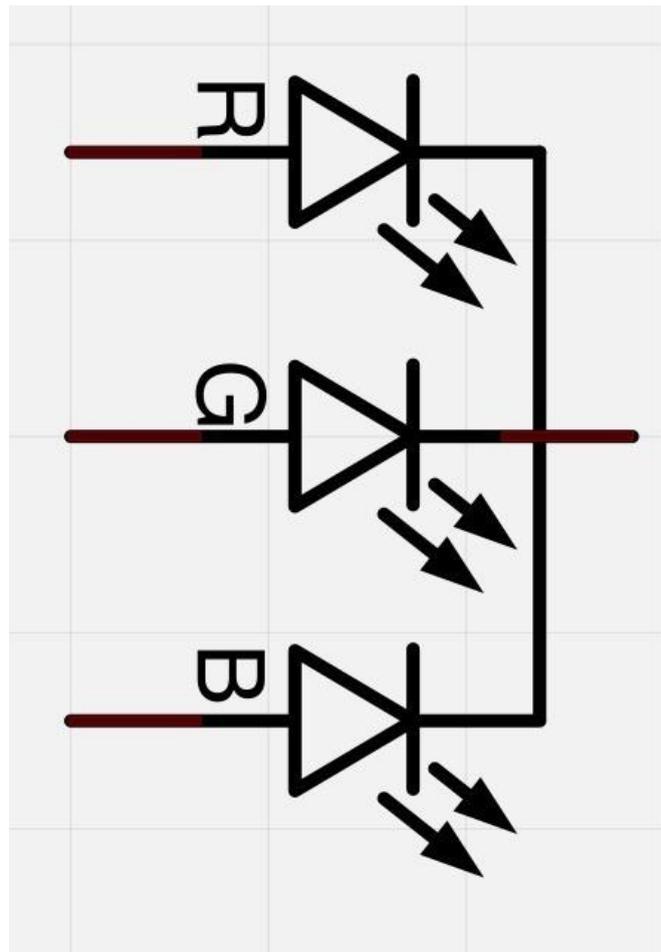
RGB:

Auf den ersten Blick sehen RGB (Rot, Grün, Blau) LEDs wie normale LEDs aus. Innerhalb der RGB-LED befinden sich jedoch drei eigentliche LEDs: eine rote, eine grüne und eine blaue. Durch Anpassen der Helligkeit jeder der drei Farben und zusammenmischen dieser, lassen sich so gut wie alle Farben produzieren.

Die Farben werden dabei ähnlich wie Farbe auf einer Farbpalette gemischt – durch Anpassen der Menge der Grundfarben bzw der drei LEDs. Der schwerste Weg dies zu erreichen wäre durch das Benutzen verschiedener Widerstände oder regelbarer Widerstände, wie wir es bereits in Lektion 2 getan haben, aber das bringt einen großen Aufwand mit sich. Glücklicherweise hat das UNO R3 Board eine *analogWrite*-Funktion, die einem erlaubt die Spannung an den analogen Pins des Boards (mit $a\sim$ markiert) anzupassen.

Die RGB-LED hat vier Kontakte. Es gibt drei Kontakte, die jeweils zum positiven Ende der drei Farb-LEDs gehen und ein Kontakt, der gemeinsam von allen Farben als negativer Anschluss genutzt wird (gemeinsame Kathode).





Hier auf den Bildern sehen Sie eine 4-Elektroden-LED. Die positiven Anschlüsse für Grün, Blau und Rot werden *Anoden* genannt. An diese muss immer der +Pol angeschlossen werden. Die Kathode wird dagegen immer mit GND (= Ground = Erdung = Minus-Pol) verbunden. Wenn Sie die LED andersherum anschließen, wird sie nicht aufleuchten.

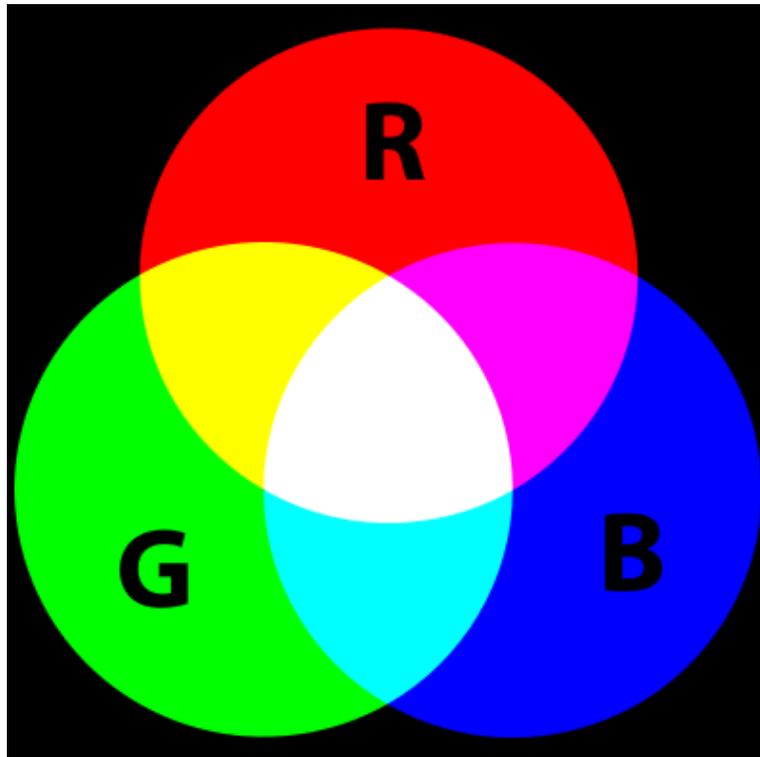
Normalerweise ist der gemeinsame negative Anschluss (gemeinsame Kathode) der zweite Pin von der flachen Seite aus. Es ist außerdem der größte aller vier Pins und wird mit GND verbunden. An jedem der drei positiven Anschlüsse muss ein 220Ω Widerstand vorgeschaltet werden, um den Strom der LED zu begrenzen.

Die anderen Enden der Widerstände müssen mit dem UNO Board verbunden werden, sodass die LEDs mit dem Board verbunden sind.

Farbe:

Der Grund, warum man durch Anpassen der Helligkeit von rot, grün und blau jede beliebige Farbe mischen kann, ist, dass das menschliche Auge drei verschiedene Lichtrezeptoren (rot, grün und blau) hat. Das Gehirn verarbeitet die Menge von rot, grün und blau und mischt sie zu einer Farbe aus dem Farbspektrum zusammen.

Wir nutzen diesen Trick also aus, um künstlich verschiedene Farben durch Anpassen der rot- grün- und blau-Werte zu erzeugen. Das RGB-Farbmodell wird beispielsweise außerdem in Fernsehern benutzt, bei denen jeder Pixel auf dem LCD-Panel aus drei Farben (rot, grün und blau) besteht.



Wenn wir die Helligkeit aller drei LEDs auf den gleichen Wert einstellen, wird am Ende weiß herauskommen. Wenn wir dann die Helligkeit der blauen LED auf 0 schalten, wird ein gelbes Licht leuchten.

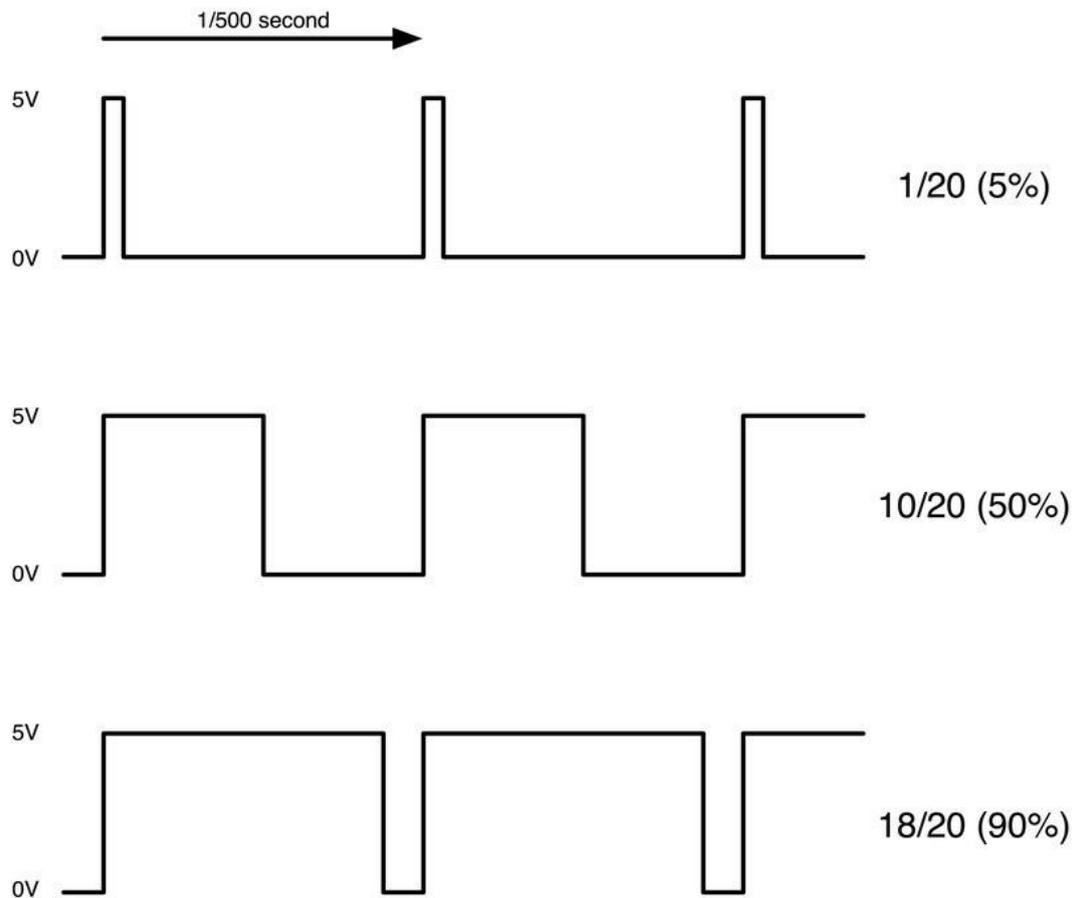
Wir können die Helligkeit der rot, grünen und blauen Teile der LED separat einstellen, was uns ermöglicht jede Farbe, die wir möchten, zusammenzumischen.

Bei Schwarz handelt es sich dagegen um keine eigentliche Farbe, da schwarz im Prinzip nur das Nichtvorhandensein von jeglichem Licht darstellt. Daher können wir mit unserer RGB-LED schwarz nur darstellen, indem wir alle drei Farben ausschalten bzw auf 0 setzen.

PWM-Technik

Die Pulsweitenmodulation (PWM) ist eine Technik, um die elektrische Leistung zu kontrollieren bzw zu begrenzen. Wir benutzen diese Technik auch, um die Helligkeit der einzelnen LEDs zu steuern.

Das Diagramm unten zeigt ein Beispiel eines PWM-Signals aus einem PWM Pin des UNO Boards.

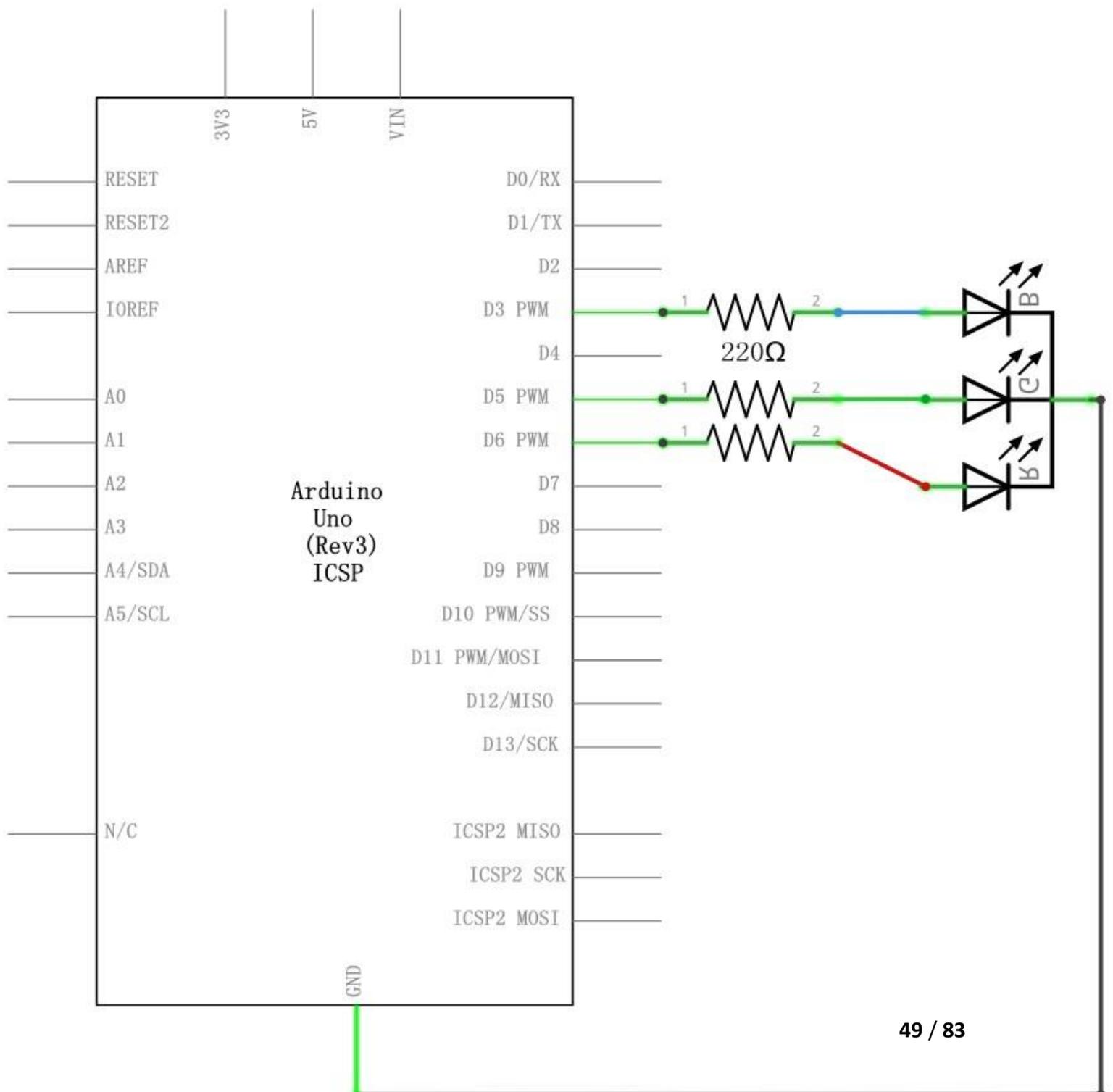


Ungefähr jedes 1/500 einer Sekunde gibt der PWM Ausgang einen Puls aus. Die Länge dieses Pulses wird durch die *analogWrite* Funktion bestimmt. So wird bei „*analogWrite(0)*“ nicht einmal ein Puls ausgegeben, während „*analogWrite(255)*“ einen Puls ausgibt, der die ganze Zeit dauert, bis der nächste Puls ansteht, sodass der Ausgang eigentlich die ganze Zeit eingeschaltet ist.

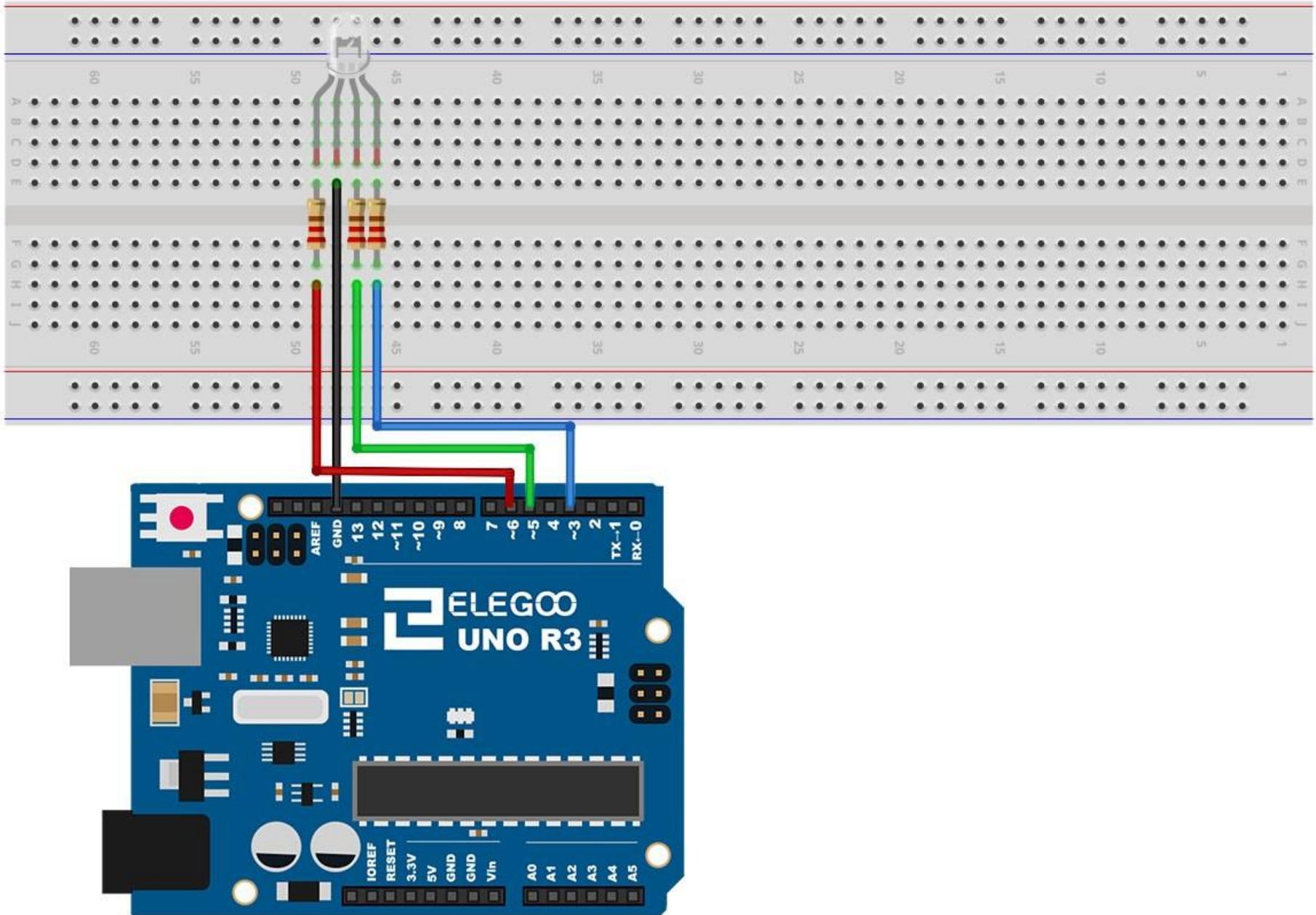
Wenn wir einen Wert irgendwo zwischen 0 und 255 für die *analogWrite*-Funktion festlegen, wird ein Puls produziert. Wenn die Länge des Pulses nur 5% der Zeit beträgt, bekommt das angeschlossene Gerät auch nur 5% der vollen Leistung.

Wenn der Ausgang bei 5V zu 90% der Zeit eingeschaltet ist, wird das Gerät 90% der vollen Leistung erhalten. Trotz des Wechsels zwischen An und Aus, können wir dieses Ein- und Ausschalten nicht an den LEDs erkennen, da unser Auge die schnell wechselnden Zustände zusammenmischt, sodass es für uns aussieht, als würde sich die Helligkeit ändern.

Verbindungsschema



Schaltplan



Code

Nach dem Verbinden der Komponenten öffnen Sie bitte den Sketch im Code-Ordner unter „Lesson 4 RGB LED“ und laden ihn auf Ihr UNO Board hoch. Bei Fragen zum Hochladen eines Sketches schauen Sie sich bitte [Lektion 2](#) noch einmal an.

Unser Code wird *FOR-Schleifen* benutzen, um durch die verschiedenen Farben zu wechseln.

Die erste FOR-Schleife wird von rot nach grün gehen.

Die zweite FOR-Schleife geht von grün nach blau.

Die letzte FOR-Schleife geht schließlich von blau zurück nach rot.

Probieren Sie den Sketch einmal aus, bevor wir ihn detaillierter besprechen.

Der Sketch fängt damit an, dass erstmal bestimmt wird, welche Pins für die verschiedenen Farben genutzt werden.

```
// Define Pins
#define BLUE 3
#define GREEN 5
#define RED 6
```

Der nächste Schritt ist die setup-Funktion. Wie wir in den vorherigen Lektionen gelernt haben, wird die setup-Funktion immer nach einem Reset des Arduinos ausgeführt. In diesem Fall ist die einzige Aufgabe der Funktion das Einstellen der Modi der Pins, die wir als Ausgänge für die LEDs benutzen werden.

```
void setup()
{
  pinMode(RED, OUTPUT);
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);
  digitalWrite(RED, HIGH);
  digitalWrite(GREEN, LOW);
  digitalWrite(BLUE, LOW);
}
```

Bevor wir uns die loop-Funktion anschauen, lassen Sie uns zuerst einen Blick auf die letzte Funktion des Sketches werfen.

Die Variablen definieren

```
redValue = 255; // choose a value between 1 and 255 to change the color.
```

```
greenValue = 0;
```

```
blueValue = 0;
```

Diese Funktion hat drei Argumente, eins für die Helligkeit der roten, eins für die Helligkeit der grünen und eins für die Helligkeit der blauen LED. In jedem möglichen Fall werden die Werte Zahlen im Bereich von 0 bis 255 sein, wobei 0 ausgeschaltet und 255 maximale Helligkeit bedeutet. Die Funktion ruft dann analogWrite auf, um die Helligkeitswerte anschließend auch zu setzen.

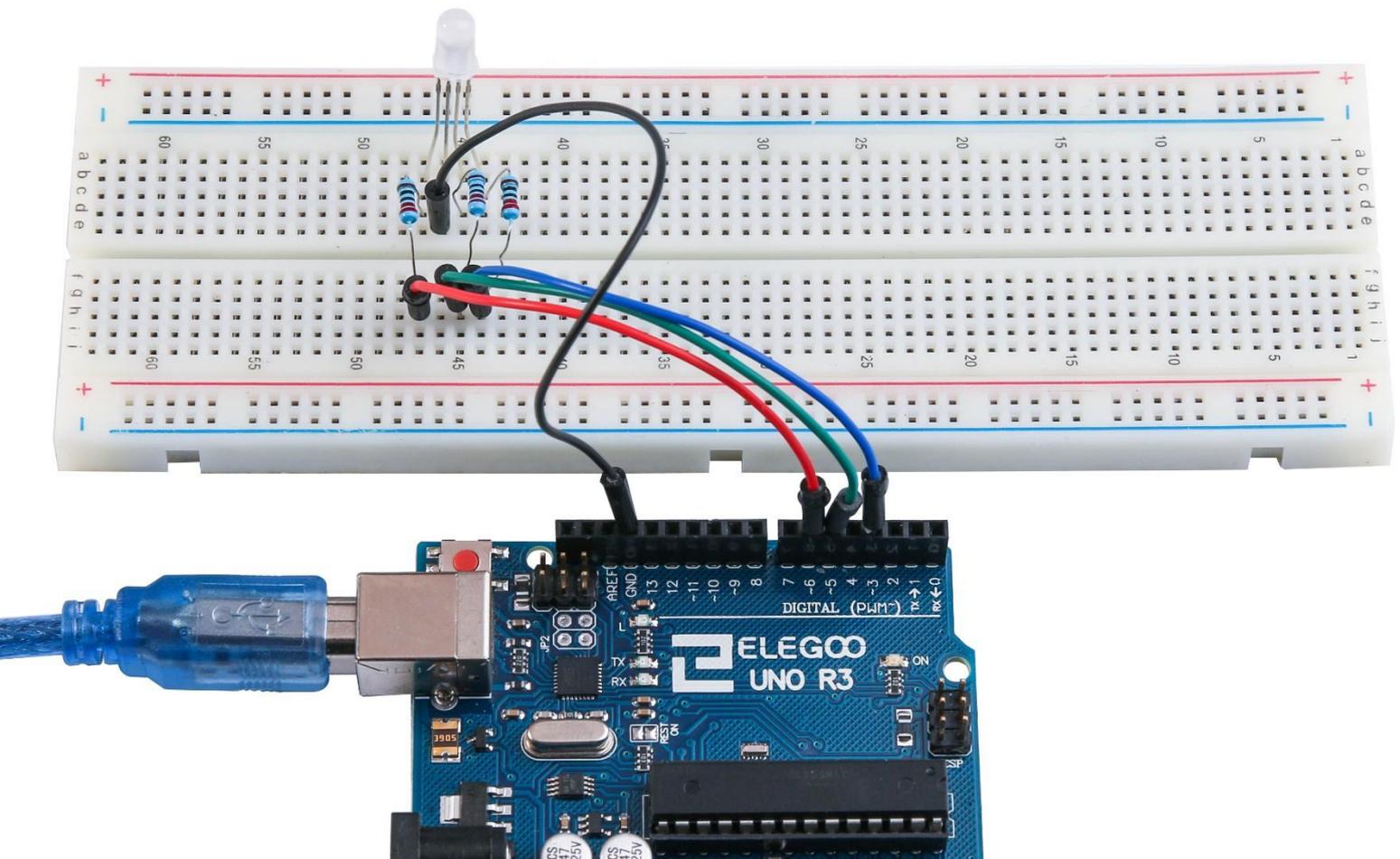
Wenn Sie nun auf die loop-Funktion schauen, können Sie sehen, dass wir die Helligkeiten von rotem, grünem und blauem Licht einstellen und dann für eine Sekunde pausieren, bevor es weiter zur nächsten Farbe geht.

```
#define delayTime 10 // fading time between colors
```

```
Delay(delayTime);
```

Versuchen Sie selber einige Farben aus den drei Grundfarben zu programmieren und schauen Sie sich den Effekt auf der LED an, wenn sie am leuchten ist.

Beispielbild



Lektion 5: Digitale Eingänge

Übersicht

In dieser Lektion werden Sie lernen Drucktaster in Verbindung mit den digitalen Eingängen des UNO Boards zu benutzen, um eine LED an- und auszuschalten.

Durch Drücken des einen Schalters wird die LED eingeschaltet, drückt man den anderen Schalter, wird sie wieder ausgeschaltet.

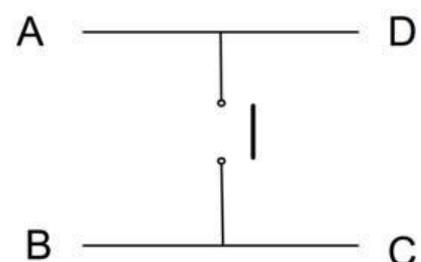
Benötigte Bauteile:

- (1) x Elegoo UNO R3
- (1) x 830 Punkte Breadboard
- (1) x 5mm rote LED
- (1) x 220 Ohm Widerstand
- (2) x Drucktaster
- (7) x M-M Kabel (Männlich zu Männlich DuPont Jumper Kabel)

Einführung in die Komponenten

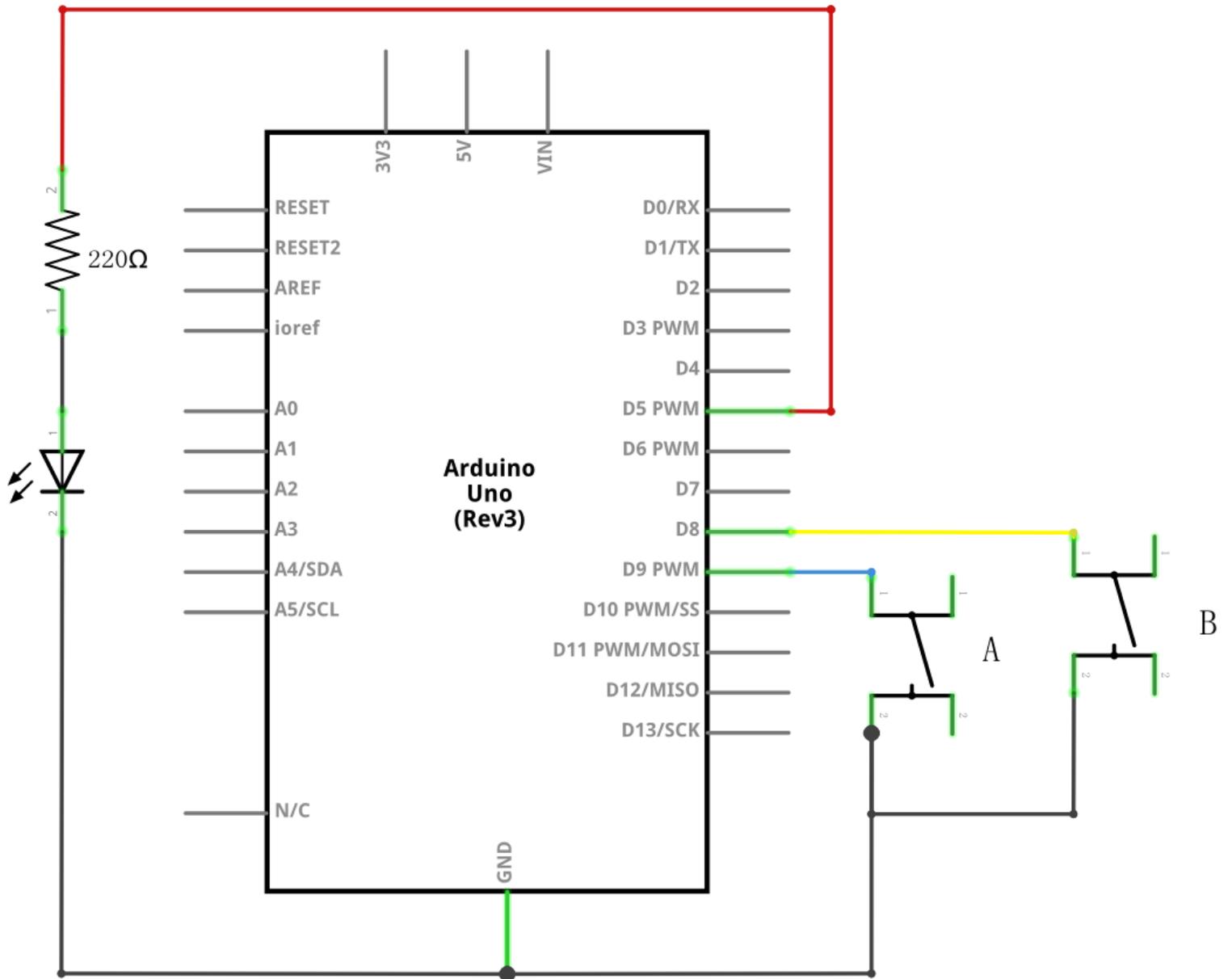
Drucktaster:

Schalter sind sehr einfache Bauteile. Wenn Sie einen Knopf drücken oder einen Hebel umlegen, verbinden sich zwei Kontakte miteinander, sodass elektrischer Strom fließen kann. Die kleinen Drucktaster, die wir in dieser Lektion benutzen werden, haben vier Kontakte, die zuerst kompliziert erscheinen.

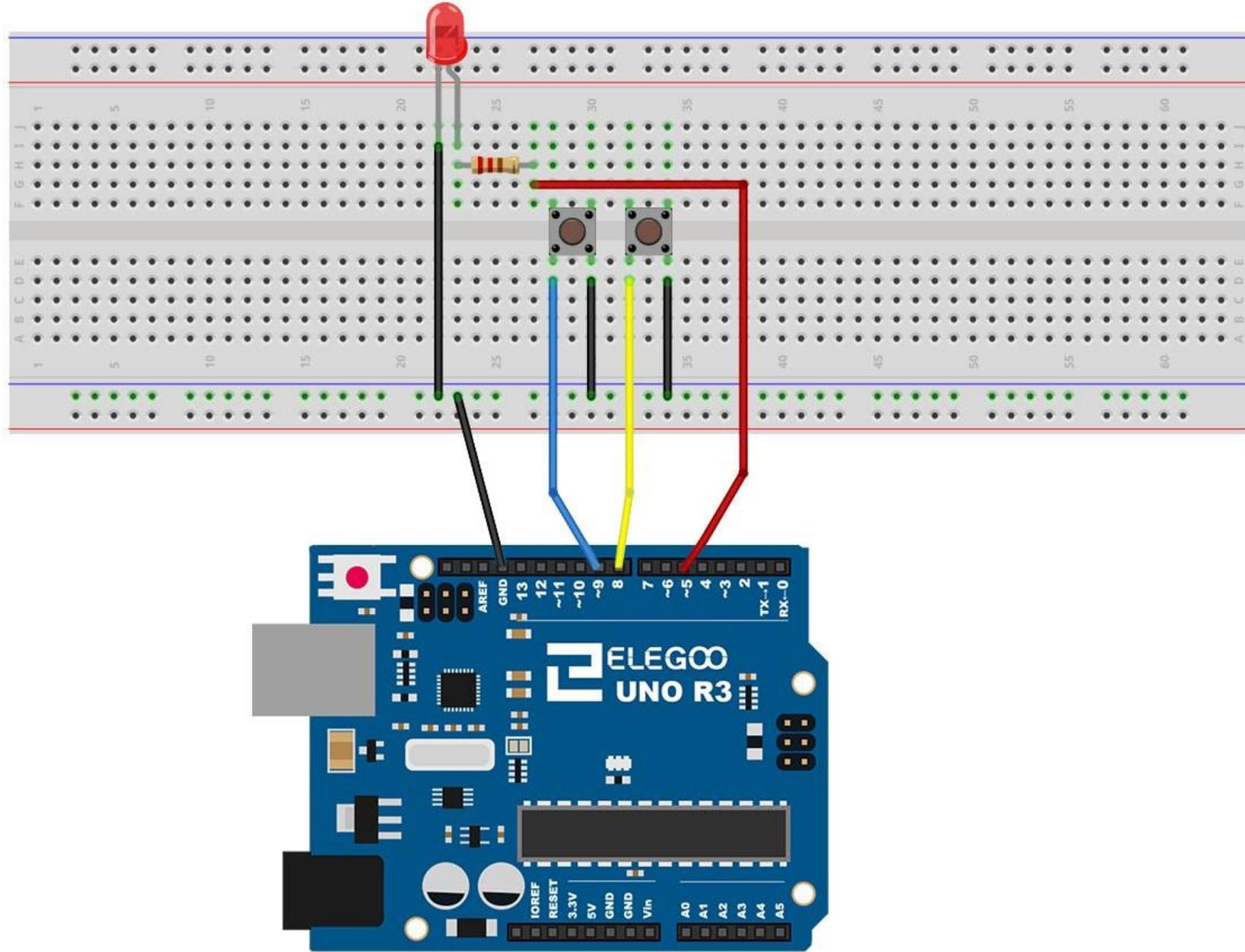


Eigentlich gibt es nur zwei richtige elektrische Verbindungen. Innerhalb des Schalters sind die Pins A und D sowie die Pins B und C miteinander verbunden.

Verbindungsschema



Schaltplan



Obwohl die Schalter quadratisch sind, können sie nicht in jede Richtung aufgesteckt werden. Das bedeutet, dass die Pins richtig auf dem Breadboard platziert werden müssen, damit sie in die Mitte des Breadboards passen.

Denken Sie daran, dass die LED den positiven Anschluss auf der rechten Seite hat.

Code

Nach dem Verbinden der Komponenten öffnen Sie bitte den Sketch im Code-Ordner unter „Lesson 5 Digital Inputs“ und laden ihn auf Ihr UNO Board hoch. Bei Fragen zum Hochladen eines Sketches schauen Sie sich bitte Lektion 2 noch einmal an.

Laden Sie den Sketch auf Ihr UNO Board. Durch Drücken des linken Schalters wird die LED eingeschaltet, ein Druck auf den rechten Schalter schaltet sie wieder aus.

Der erste Teil des Sketches definiert drei Variablen für die drei Pins, die benutzt werden. Der „*ledPin*“ ist der Ausgang für die LED, der „*buttonApin*“ verweist auf den Schalter, der näher oben am Breadboard liegt und der „*buttonBpin*“ steht für den unteren Schalter.

Die setup-Funktion definiert den ledPin als einen gewöhnlichen Ausgang (OUTPUT), wohingegen die beiden Schalterpins als Eingänge behandelt werden müssen. In diesem Fall setzen wir den pin-Modus auf „*INPUT_PULLUP*“:

```
pinMode(buttonApin, INPUT_PULLUP);  
pinMode(buttonBpin, INPUT_PULLUP);
```

Der pin-Modus INPUT_PULLUP bedeutet, dass der Pin als Eingang behandelt werden soll. Gleichzeitig bestimmt er, dass wenn nichts an den Eingang angeschlossen ist, der Standardwert für den Eingang „*HIGH*“ ist. In anderen Worten: Der Standardwert für den Pin ist HIGH, außer er wurde durch einen betätigten Schalter auf „*LOW*“ geschaltet.

Das ist der Grund, warum die Schalter mit GND verbunden sind. Wenn ein Taster gedrückt wurde, verbindet er den Eingangs-Pin mit GND, sodass der Eingang nicht länger HIGH geschaltet ist. Da der Eingang im Grundzustand HIGH geschaltet und bei gedrücktem Schalter LOW ist, ist die Logik umgekehrt als eigentlich üblich. Darum werden wir uns in der loop-Funktion kümmern.

```

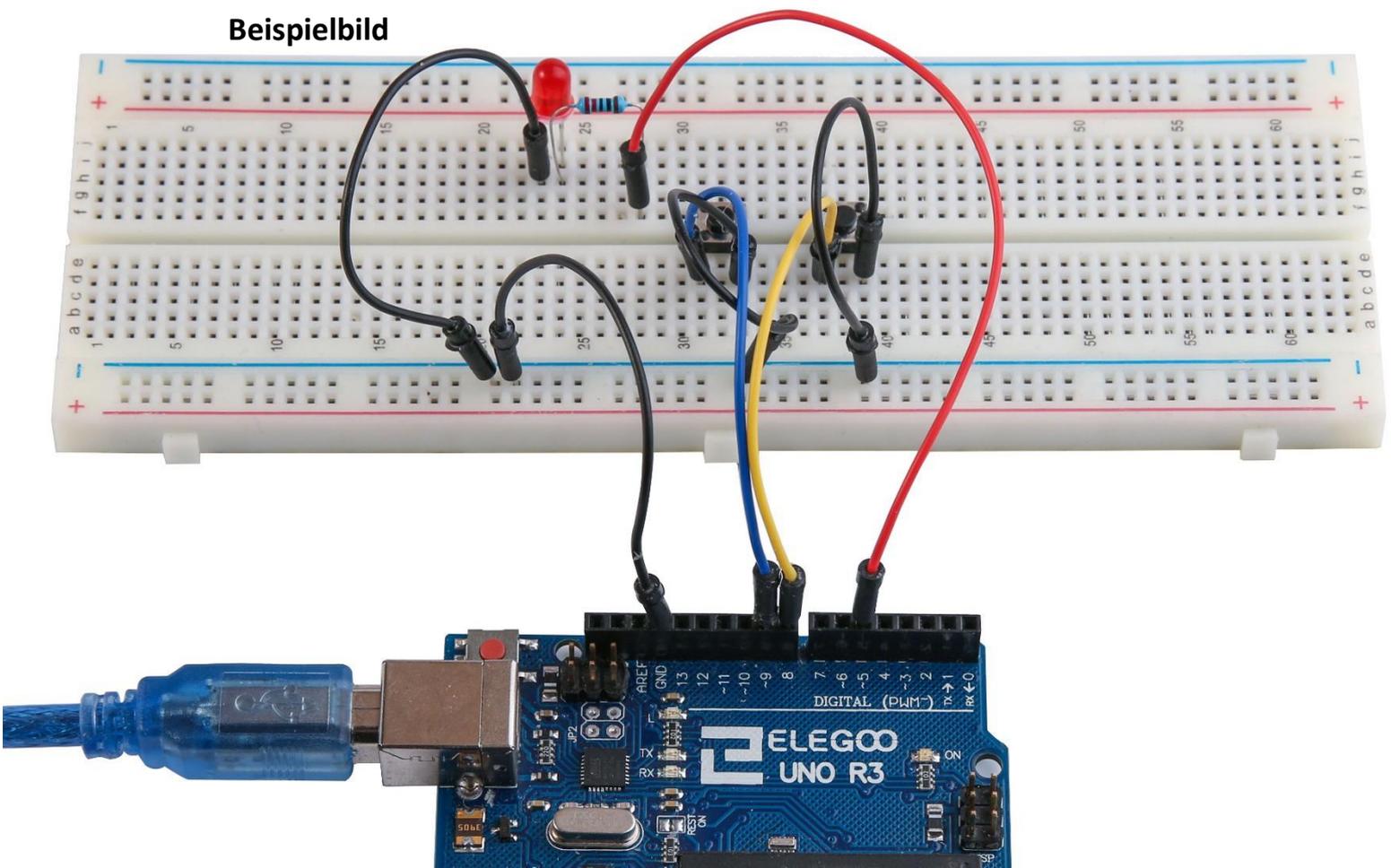
void loop()
{
  if (digitalRead(buttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  if (digitalRead(buttonBpin) == LOW)
  {
    digitalWrite(ledPin, LOW);
  }
}

```

In der loop-Funktion befinden sich zwei „if“-Statements (Bedingungsanweisungen). Für jeden Schalter eins. Beide if-Statements haben als Bedingung ein „*digitalRead*“, das den Zustand des jeweiligen Pins beobachtet.

Bedenken Sie, dass bei jedem Druck eines Tasters der Wert dessen Pins LOW sein wird. Wenn der Taster A gedrückt wird, wird *buttonApin* LOW sein. Das erste if-Statement (*digitalRead(buttonApin) == LOW*) wird damit erfüllt sein und die LED zum leuchten bringen (*digitalWrite(ledpin, HIGH)*). Ähnlich funktioniert es beim zweiten Taster. Wenn Taster B gedrückt wird, erkennt dies das zweite If-Statement (*digitalRead(buttonBpin) == LOW*) und schaltet die LED auf LOW (= aus).

Beispielbild



Lektion 6: Aktiver Buzzer

Übersicht

In dieser Lektion lernen Sie, wie Sie einen Ton aus einem aktiven Buzzer ausgeben.

Benötigte Bauteile:

(1) x Elegoo UNO R3

(1) x Aktiver Buzzer

(2) x W-M Kabel (Weiblich zu Männlich DuPont Jumper Kabel)

Einführung in die Komponenten

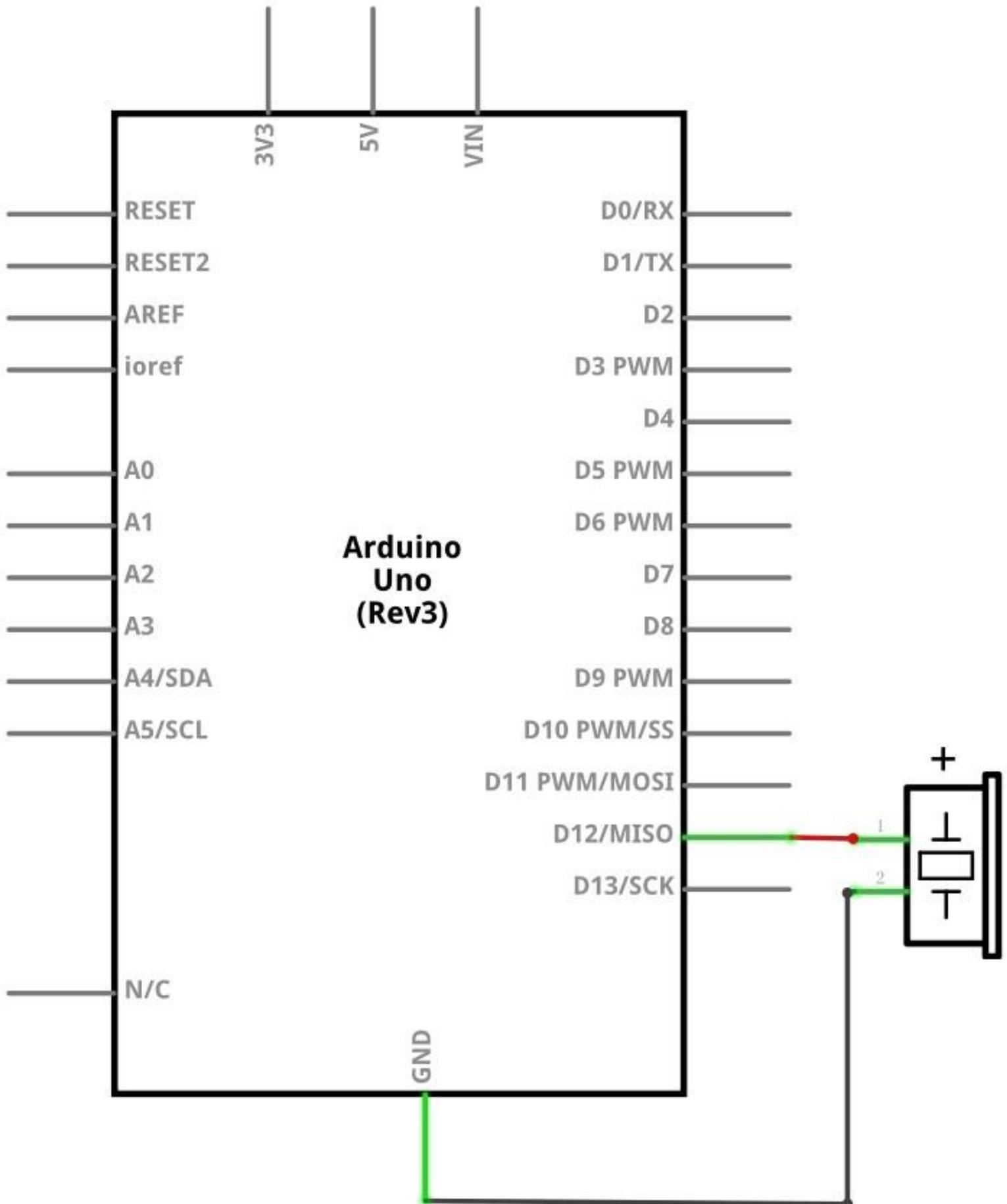
BUZZER:

Elektronische Buzzer (Summer) werden mit Gleichstrom betrieben und kommen mit einem integrierten Schaltkreis. Sie werden häufig in Computern, Druckern, Kopierern, Alarmsystemen, elektronischen Spielzeugen, Automobilelektronik, Telefonen, Timern und anderen elektronischen Geräten benutzt, um Töne auszugeben. Man kann zwischen aktiven und passiven Buzzern unterscheiden. Wenn man einen aktiven mit einem passiven Buzzer vergleicht, stellt man fest, dass der mit dem grünen PCB (Platine) der passive ist, während der mit dem schwarzen PCB der aktive Buzzer ist.

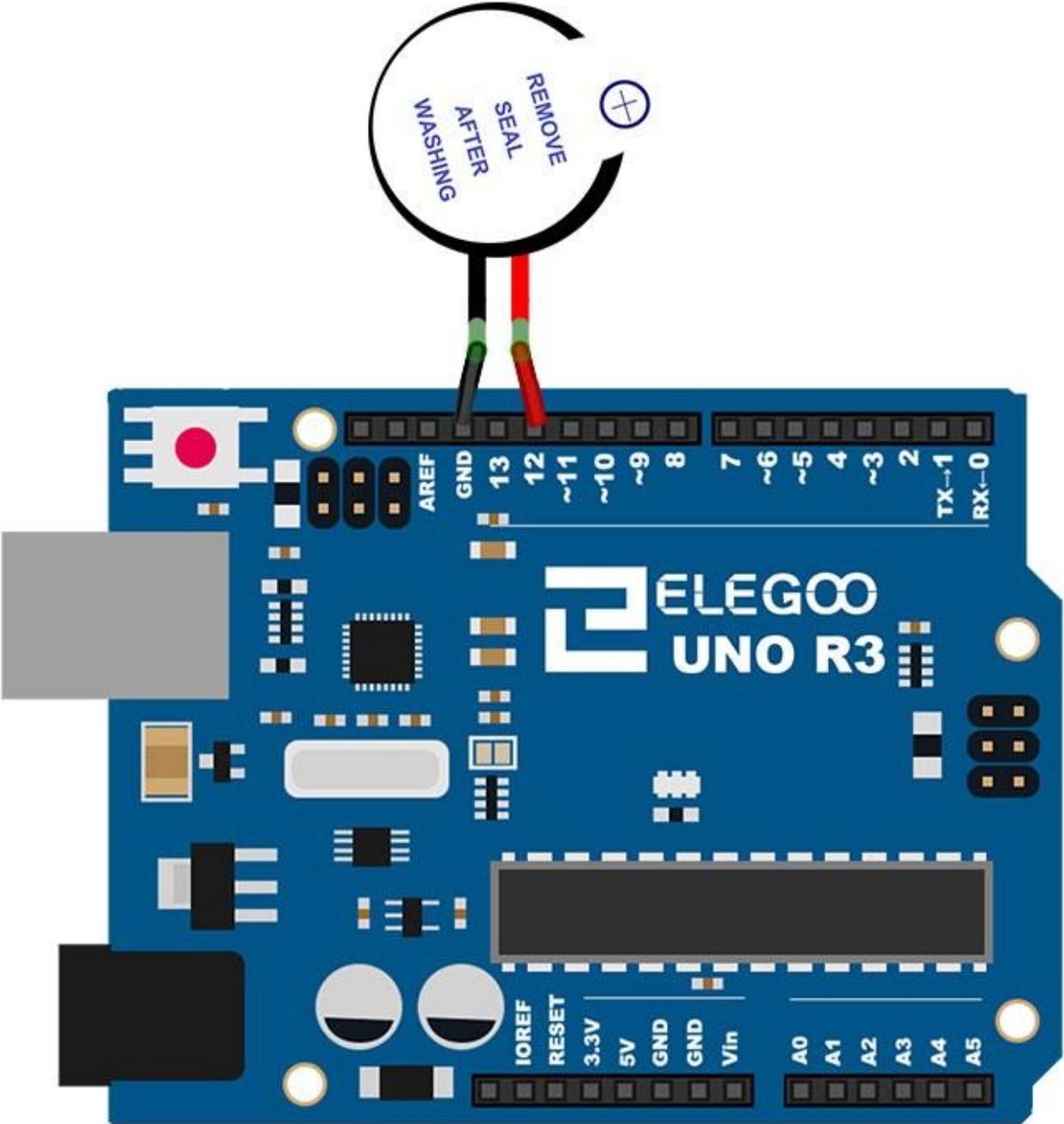
Der Unterschied zwischen den beiden Arten ist, dass der aktive Buzzer eine eingebaute Signalquelle hat, die automatisch einen Ton erzeugt, wenn Gleichstrom angelegt wird. Ein passiver Buzzer hat keine solche Signalquelle und wird daher nicht ertönen, wenn man eine Gleichstromquelle direkt anschließt. Stattdessen muss der passive Buzzer mit Rechteckschwingungen mit einer Frequenz zwischen 2000 und 5000Hz versorgt werden. Ein aktiver Buzzer ist wegen der eingebauten Schaltkreise oft teurer als ein passiver.



Verbindungsschema



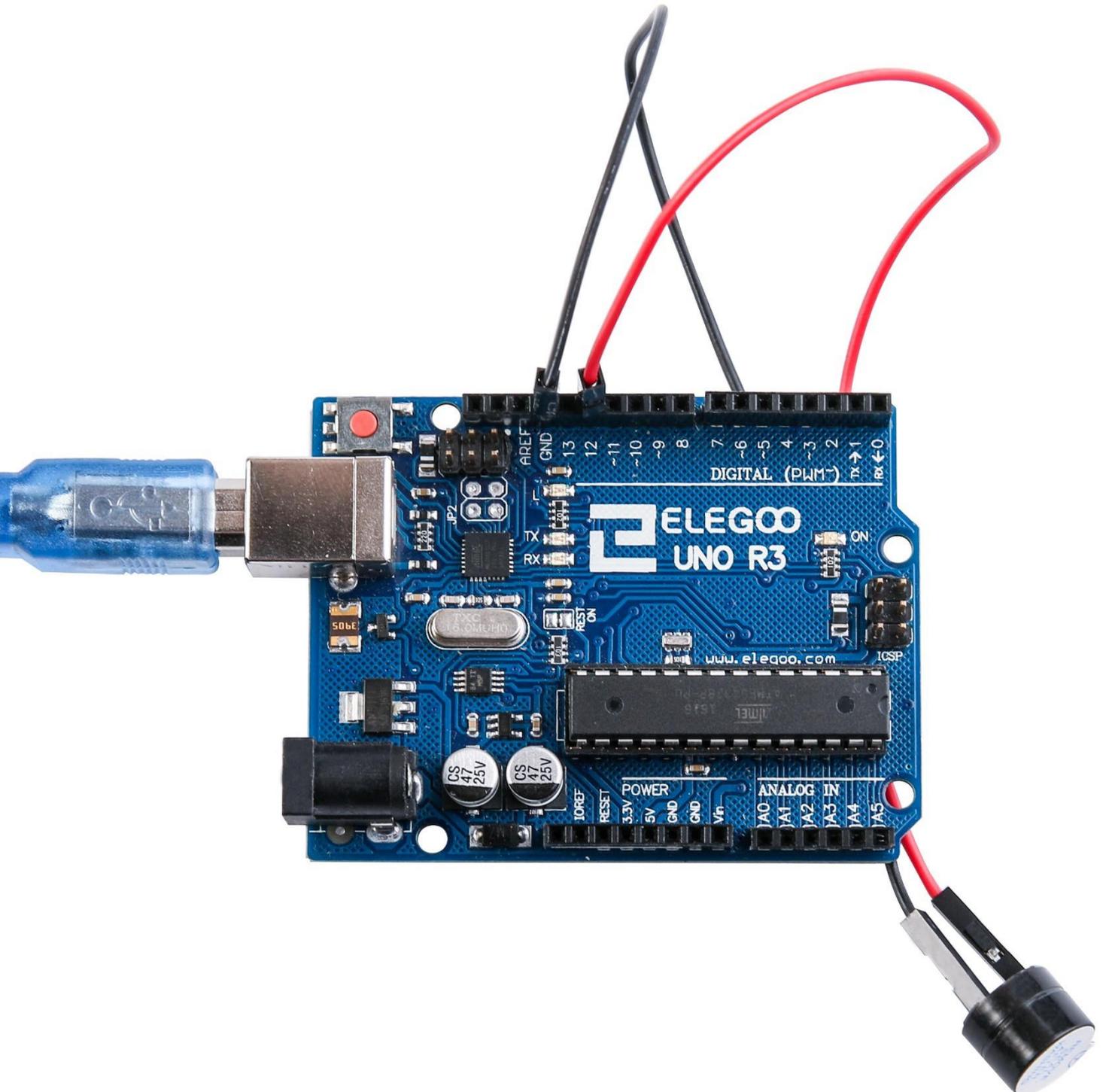
Schaltplan



Code

Nach dem Verbinden der Komponenten öffnen Sie bitte den Sketch im Code-Ordner unter „Lesson 6 Making Sounds“ und laden ihn auf Ihr UNO Board hoch. Bei Fragen zum Hochladen eines Sketches schauen Sie sich bitte Lektion 2 noch einmal an.

Beispielbild



Lektion 7 Neigungssensor

Übersicht

In dieser Lektion lernen sie, wie man einen Neigungssensor benutzt, um den Grad der Neigung herauszufinden.

Benötigte Bauteile:

- (1) x Elegoo UNO R3
- (1) x Neigungssensor
- (2) x W-M Kabel (Weiblich zu Männlich DuPont Jumper Kabel)



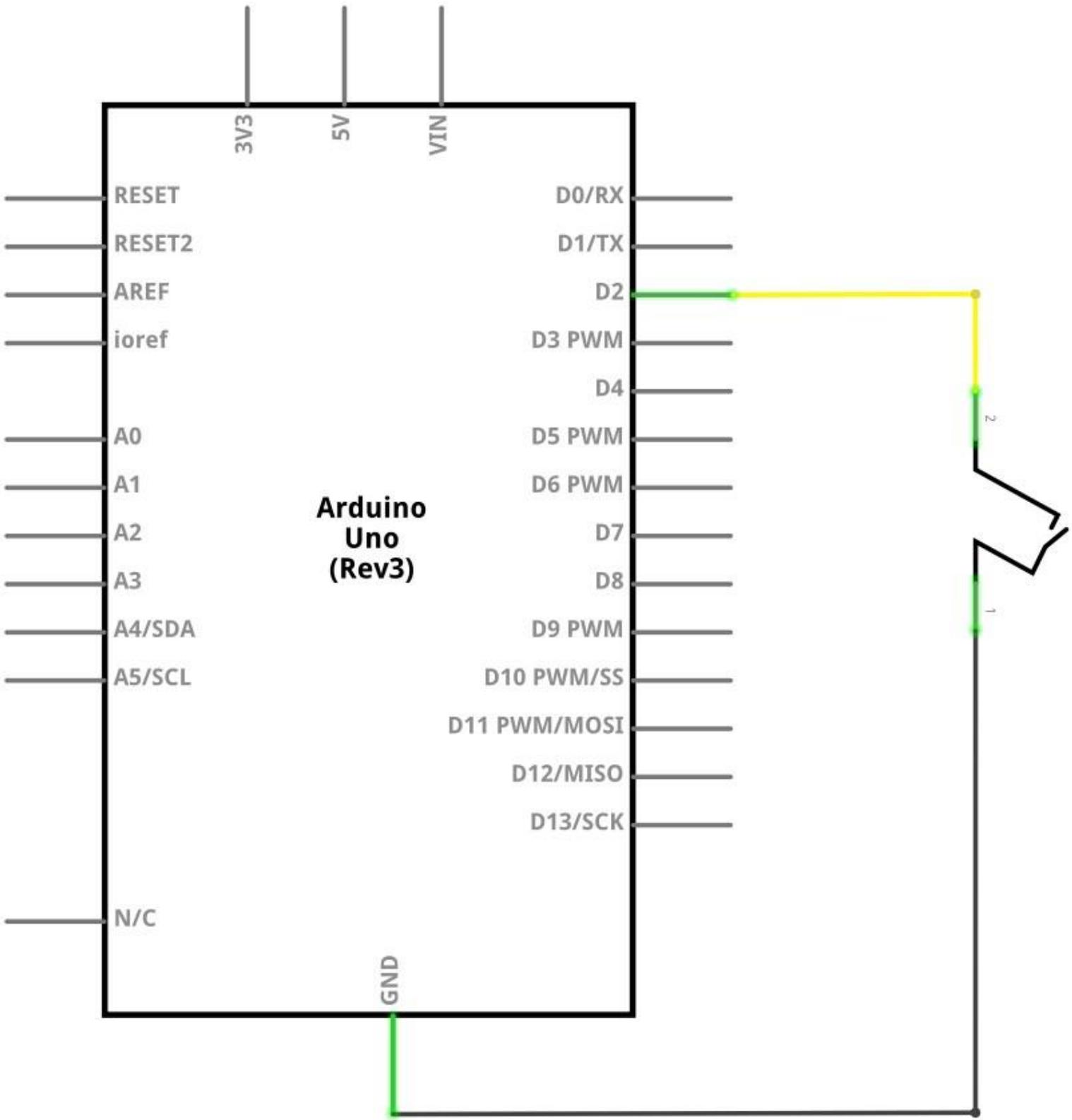
Einführung in die Komponenten

Neigungssensor:

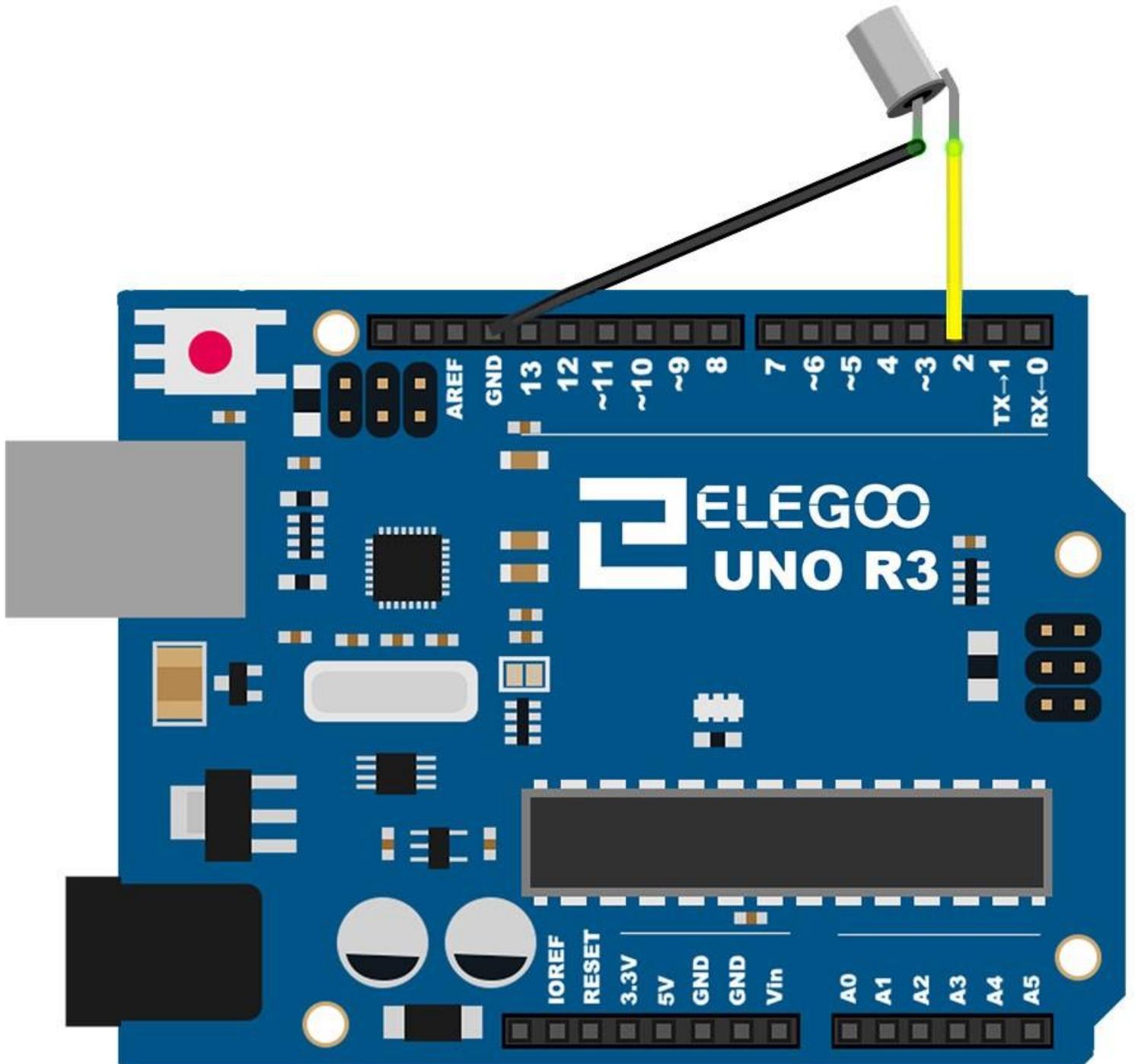
Neigungssensoren erlauben einem die momentane Ausrichtung oder Neigung festzustellen. Sie sind klein, günstig, einfach zu benutzen und verbrauchen wenig Strom. Wenn sie richtig benutzt werden, halten sie ewig. Durch ihren simplen Aufbau werden sie häufig in Spielzeugen, Gadgets und anderen Geräten eingesetzt. Sie bestehen in der Regel aus einer Art Hohlraum (häufig in Form eines Zylinders) mit einer leitfähigen beweglichen Masse innerhalb. An einem Ende des Hohlraums befinden sich zwei leitende Elemente (Pole). Wenn der Neigungssensor senkrecht, wie im Bild oben, ausgerichtet ist, rollt die bewegliche Masse auf die beiden Pole und verbindet sie miteinander, wie bei einem Schalter.

Obwohl sie nicht so präzise und flexibel wie Beschleunigungssensoren sind, können Neigungssensoren Erschütterungen erkennen oder die grobe Neigung bestimmen. Ein weiterer Vorteil ist, dass große Neigungssensoren direkt die Stromversorgung schalten können. Beschleunigungssensoren geben dagegen digitale oder analoge Spannungswerte aus, die erst durch einen weiteren Schaltkreis oder Mikrocontroller analysiert werden müssen.

Verbindungsschema



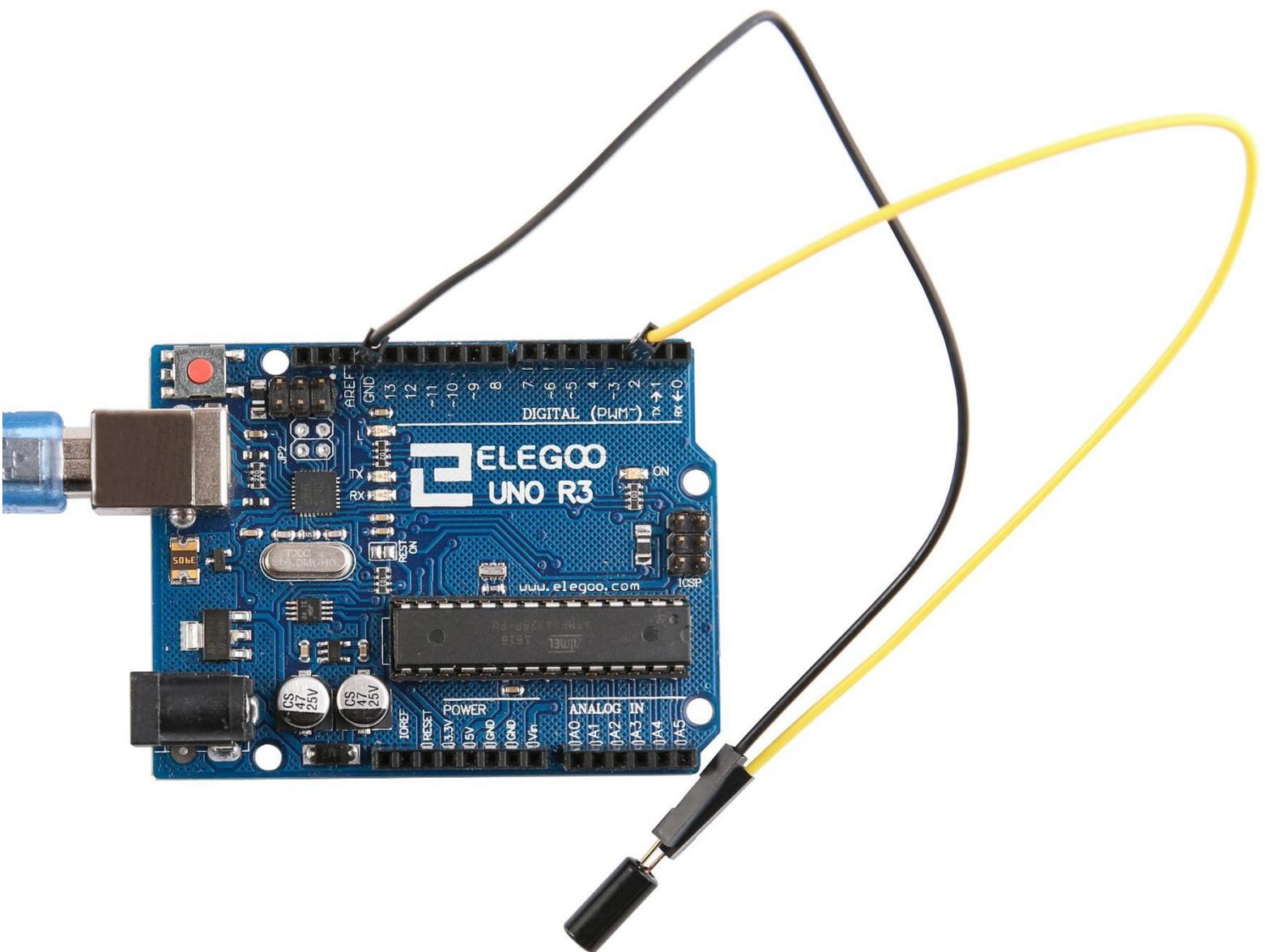
Schaltplan



Code

Nach dem Verbinden der Komponenten öffnen Sie bitte den Sketch im Code-Ordner unter „Lesson 7 Ball Switch“ und laden ihn auf Ihr UNO Board hoch. Bei Fragen zum Hochladen eines Sketches schauen Sie sich bitte Lektion 2 noch einmal an.

Beispielbild



Lektion 8 Acht LEDs per 74HC595 ansteuern

Übersicht

In dieser Lektion lernen Sie, wie man acht rote LEDs mit dem UNO Board verbinden kann, ohne dafür 8 Pins belegen zu müssen.

Sie könnten die acht LEDs auch einzeln mit vorgeschaltetem Widerstand mit dem UNO Board verbinden, Ihnen würden dabei aber sehr schnell die Pins ausgehen. Wenn Sie nur wenige Komponenten mit Ihrem Board verbinden möchten, funktioniert das noch, aber meistens möchte man noch weitere Schalter, Sensoren, Servos usw. anschließen und früher oder später gehen Ihnen die Anschlüsse aus, weil alle belegt sind. Also statt alle LEDs einzeln anzuschließen, werden wir in dieser Lektion den 74HC595 Chip benutzen. Der 74HC595 ist ein Seriell zu Parallel Wandler. Der Chip hat acht Ausgänge (was für unser Vorhaben perfekt ist) und drei Eingänge, über die er mit Daten versorgt werden kann.

Durch den Chip lassen sich die LEDs ein bisschen langsamer ansteuern, so kann man die LEDs nur noch 500.000 pro Sekunde umschalten statt 8.000.000 pro Sekunde, wie es ohne Chip der Fall wäre. Das ist nichtsdestotrotz immer noch enorm schnell, sodass man den Unterschied mit dem Auge niemals wahrnehmen wird.

Benötigte Bauteile:

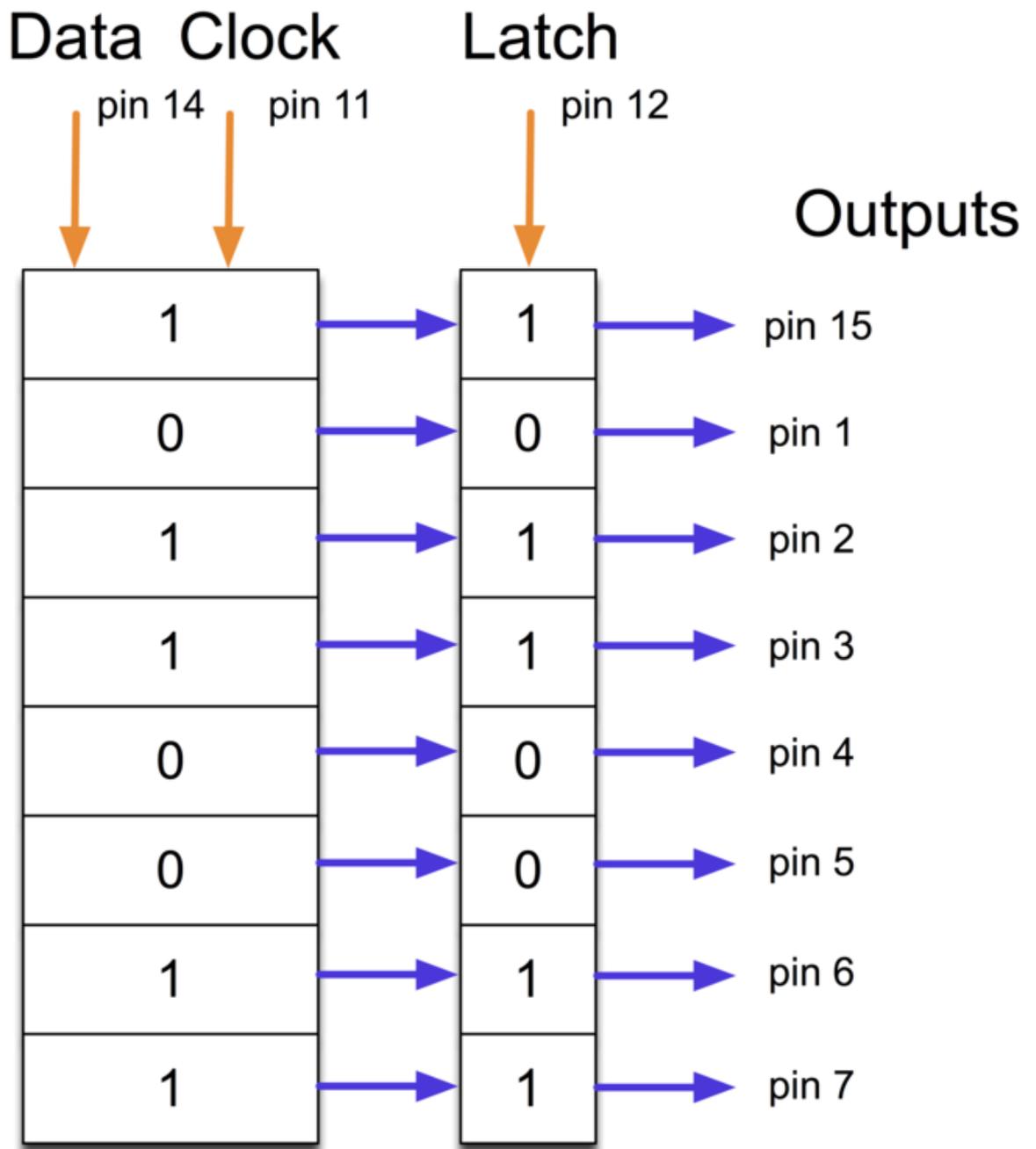
- (1) x Elegoo UNO R3
- (1) x 830 Punkte Breadboard
- (8) x Rote LEDs
- (8) x 220 Ohm Widerstände
- (1) x 74HC595 IC
- (14) x M-M Kabel (Männlich zu Männlich DuPont Jumper Kabel)



Einführung in die Komponenten

74HC595 Schieberegister:

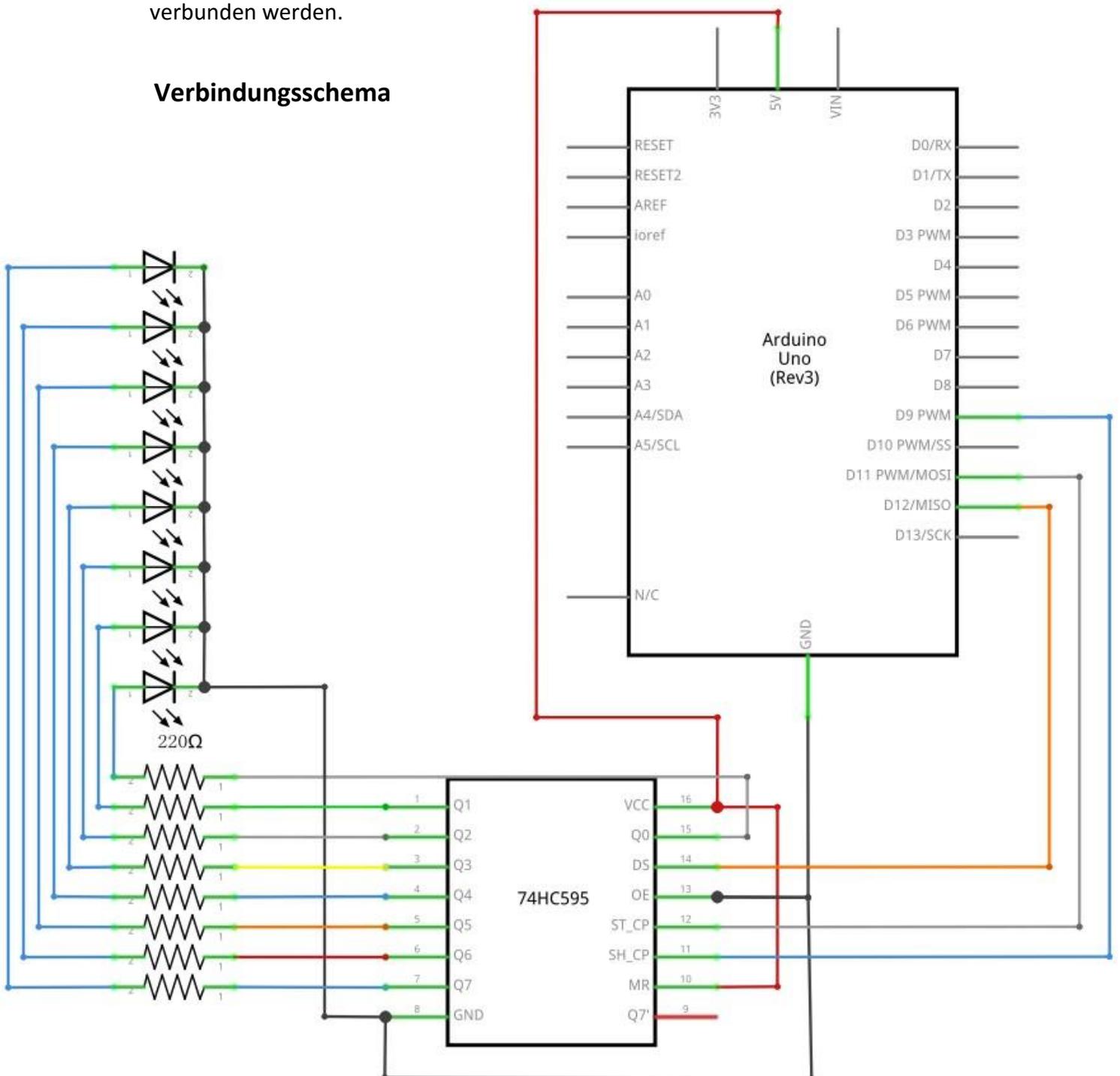
Ein Schieberegister ist ein Chip, der acht Speicherstellen hat, die jeweils entweder eine 1 oder eine 0 speichern können. Um die Werte zu verändern, werden wir die Daten über den „Data“- und den „Clock“-Anschluss des Chips senden.



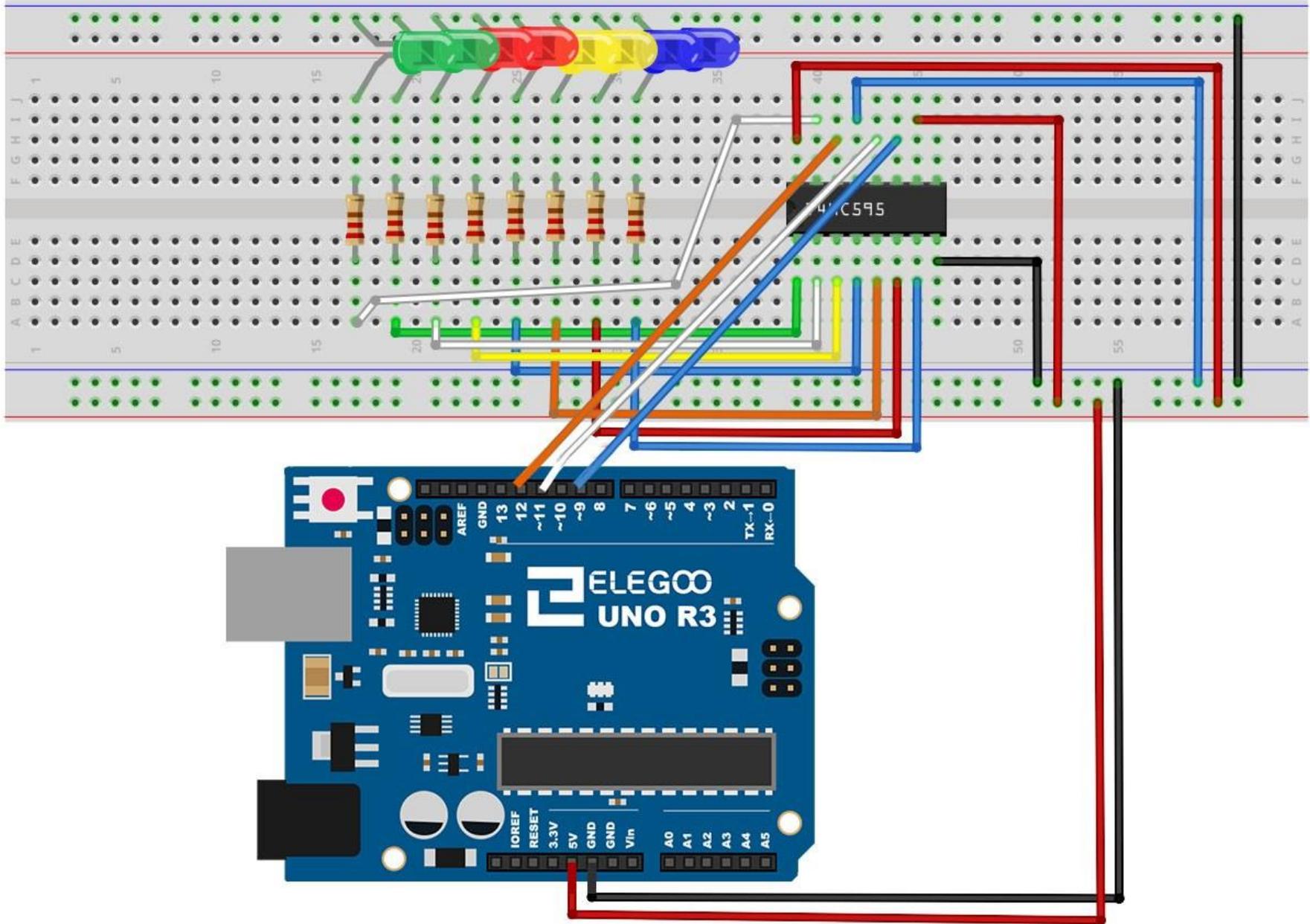
An den Clock-Pin müssen acht Pulse geschickt werden. Bei jedem Puls wird der Data-Pin überprüft. Wenn der Data-Pin HIGH geschaltet ist, wird eine 1 in das Schieberegister an die jeweilige Stelle geschrieben. Wenn der Pin LOW ist, wird eine 0 geschrieben. Nachdem alle 8 Pulse empfangen wurden, muss der „Latch“-Pin aktiviert werden und die acht Werte werden im Register gespeichert. Das ist unbedingt notwendig, da sonst die Daten an die falschen Stellen geschrieben werden könnten und die falschen LEDs aufleuchten würden.

Der Chip hat zusätzlich einen „Output Enable“ (OE)-Pin, durch den alle Ausgänge auf einmal an bzw. ausgeschaltet werden können. Sie könnten diesen Pin an einen PWM-fähigen Pin des Boards anschließen und mit der analogWrite-Funktion die Helligkeit der LEDs kontrollieren. Der OE-Pin ist aktiv LOW, also muss er mit GND verbunden werden.

Verbindungsschema



Schaltplan



Da wir insgesamt acht LEDs und acht Widerstände anschließen müssen, müssen einige Verbindungen hergestellt werden.

Am einfachsten ist es, wenn man den 74HC595-Chip zuerst einsetzt, da so ziemlich alles mit ihm verbunden wird. Setzen sie den Chip so ein, dass die kleine Einkerbung zum oberen Ende (der kurzen Seite) des Breadboards zeigt. Der Pin 1 befindet sich am Chip unter dieser Einkerbung.

Der digitale *Pin 12* des UNOs geht an *Pin 14* des Schieberegisters, der digitale *Pin 11* des UNOs geht an *PIN 12* des Schieberegisters und der digitale *Pin 9* des UNOs geht an *PIN 11* des Schieberegisters.

Alle Ausgänge außer einem sind auf der unteren Seite des Chips. Daher werden wir die LEDs der Einfachheit halber auf der linken Seite platzieren.

Nach dem Chip müssen die Widerstände an ihre Stelle gesetzt werden. Sie müssen aufpassen, dass die Kontakte der verschiedenen Widerstände sich nicht berühren. Sie sollten dies nocheinmal überprüfen, bevor Sie das Board später mit Strom versorgen. Wenn es zu schwer ist die Widerstände einzusetzen, ohne dass sich dessen Kontakte berühren, kann es helfen die Kontakte zu kürzen, sodass die Widerstände tiefer am Breadboard anliegen.

Als nächstes platzieren Sie die LEDs auf dem Breadboard. Die langen positiven Enden müssen alle in Richtung Chip zeigen.

Schließen Sie die Jumper Kabel wie im obigen Bild gezeigt an. Vergessen Sie nicht das Kabel, das vom Pin 8 des Chips zum GND-Anschluss des Breadboards führt.

Laden Sie den Sketch hoch und probieren Sie es aus. Alle LEDs sollten nacheinander aufleuchten, bis alle an sind. Danach schalten sich alle LEDs aus und das Muster wiederholt sich anschließend.

Code

Nach dem Verbinden der Komponenten öffnen Sie bitte den Sketch im Code-Ordner unter „*Lesson 8 Eight LED with 74HC595*“ und laden ihn auf Ihr UNO Board hoch. Bei Fragen zum Hochladen eines Sketches schauen Sie sich bitte Lektion 2 nocheinmal an.

Als erstes im Sketch bestimmen wir die drei Pins, die wir benutzen werden. Dabei handelt es sich um die digitalen Ausgänge des UNO Boards, die mit den Latch-, Clock- und Data-Pins vom 74HC595 verbunden werden.

```
int latchPin = 11;
```

```
int clockPin = 9;
```

```
int dataPin = 12;
```

Als nächstes wird eine Variable mit dem Namen „*leds*“ gesetzt. Diese Variable werden wir dazu nutzen, um zu speichern welche LEDs gerade ein- und welche ausgeschaltet sind. Variablen des Typs „byte“ repräsentieren acht Bit große Zahlen. Jeder Bit kann dabei an oder aus sein. Also perfekt für die Zustände unserer LEDs.

```
byte leds = 0;
```

Die setup-Funktion bestimmt nur, dass wir unsere drei Pins als Ausgänge nutzen.

```
void setup()
{
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
}
```

Die loop-Funktion schaltet zu Beginn erst einmal alle LEDs aus, indem die *leds*-Variable auf 0 gesetzt wird. Dann wird „*updateShiftRegister*“ aufgerufen, das den Wert der *leds*-Variable an das Schieberegister sendet, was somit alle LEDs ausschaltet. Um die Funktionsweise der *updateShiftRegister*-Funktion kümmern wir uns später. Die loop-Funktion wird durch eine 0,5 sekündige Pause unterbrochen und beginnt dann in einer *for*-Schleife in Durchgängen von 0 bis 7 zu zählen, wobei die Variable *i* die aktuelle Zahl repräsentiert. Bei jedem Durchgang wird die Funktion „*bitSet*“ aufgerufen, die die Zahl in der *leds*-Variable ändert. Danach wird wieder *updateShiftRegister* aufgerufen, damit der Chip die LEDs anpasst. Dann gibt es wieder eine 0,5 sekündige Pause und danach folgt der nächste Durchgang.

```
void loop()
{
    leds = 0;
    updateShiftRegister();
    delay(500);
    for (int i = 0; i < 8; i++)
    {
        bitSet(leds, i);
        updateShiftRegister();
        delay(500);
    }
}
```

Die Funktion `updateShiftRegister` setzt zuerst den `latchPin` auf `LOW`, und ruft dann die Funktion „`shiftOut`“ auf, bevor sie den `latchPin` wieder auf `HIGH` umschaltet.

Die `shiftOut`-Funktion benötigt vier Parameter. Die ersten beiden sind die Pins, die an `Data` und `Clock` angeschlossen sind.

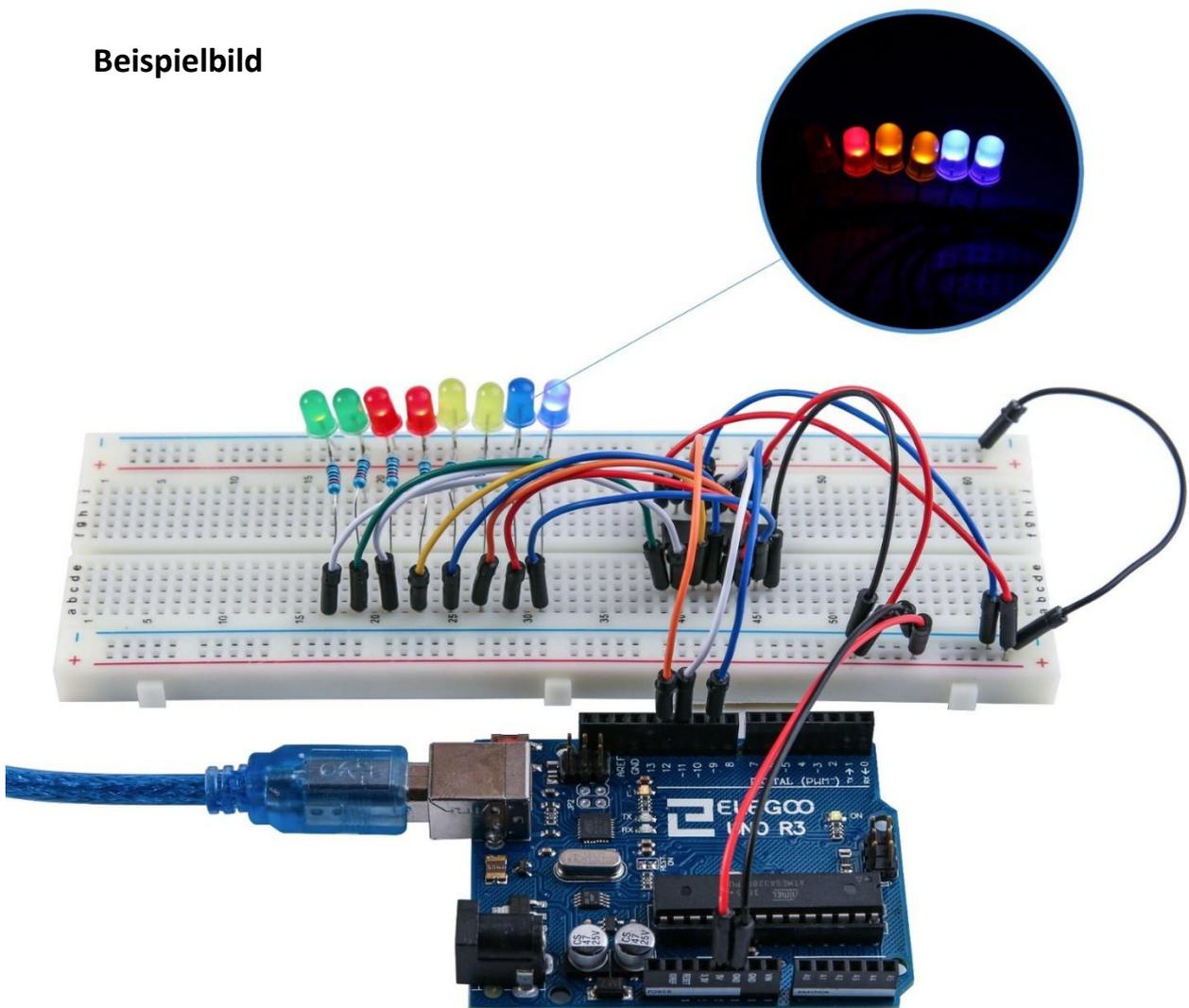
Der dritte Parameter bestimmt an welchem Ende wir starten wollen. Wir starten bei dem Bit ganz rechts, der „*Least Significant Bit*“ (*LSB*) genannt wird.

Der letzte Parameter sind die eigentlichen Daten, die wir an das Schieberegister übertragen wollen. Bei uns ist der letzte Parameter die `leds`-Variable.

```
void updateShiftRegister()  
{  
    digitalWrite(latchPin, LOW);  
    shiftOut(dataPin, clockPin, LSBFIRST, leds);  
    digitalWrite(latchPin, HIGH);  
}
```

Wenn sie stattdessen lieber eine LED aus- statt einschalten möchten, müssen sie eine ähnliche Arduino-Funktion, die „`bitClear`“-Funktion, mit der `leds`-Variable als Parameter aufrufen. Diese Funktion setzt den entsprechenden Bit auf 0. Danach müssen Sie nur noch die `updateShiftRegister`-Funktion aufrufen.

Beispielbild



Lektion 9 Der Serielle Monitor

Übersicht

In dieser Lektion werden wir auf Lektion 8 aufbauen. Wir werden die Möglichkeit einbauen die LEDs vom Computer aus über den Seriellen Monitor zu steuern. Der Serielle Monitor ist die Verbindung zwischen Ihrem Computer und dem UNO Board. Durch ihn können Sie Textnachrichten senden und empfangen, was nützlich zum debuggen und zum steuern des UNO Boards sein kann. Zum Beispiel können Sie von Ihrem Computer einen Befehl senden, der die LEDs einschalten wird. In dieser Lektion werden die selben Teile und ein ähnliches Breadboard Layout wie in Lektion 8 benutzt. Lesen Sie zuerst Lektion 8, falls Sie es noch nicht getan haben.

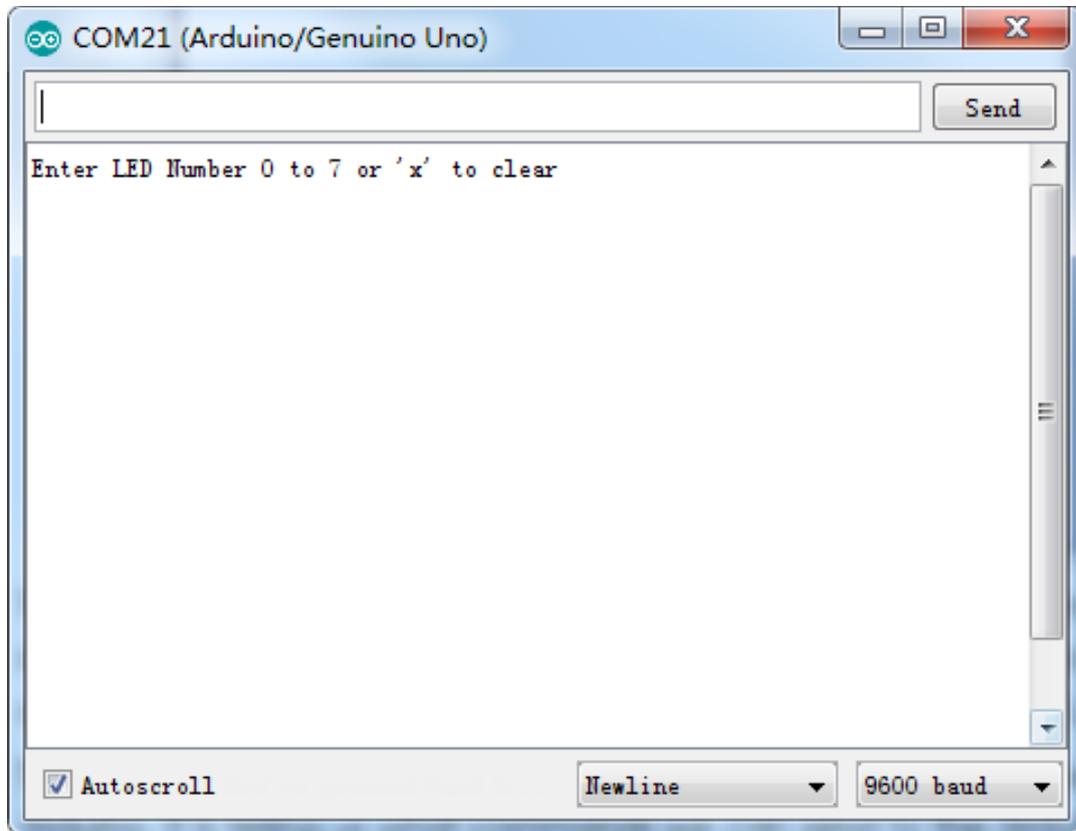
Durchführung

Nachdem Sie den Sketch auf Ihr Board hochgeladen haben, klicken Sie in der IDE auf das Symbol des Seriellen Monitors (unten in der Abbildung markiert).



Das folgende Fenster wird sich öffnen.

Die grundlegenden Informationen zum Seriellen Monitor haben Sie in [Lektion 1](#) kennengelernt.

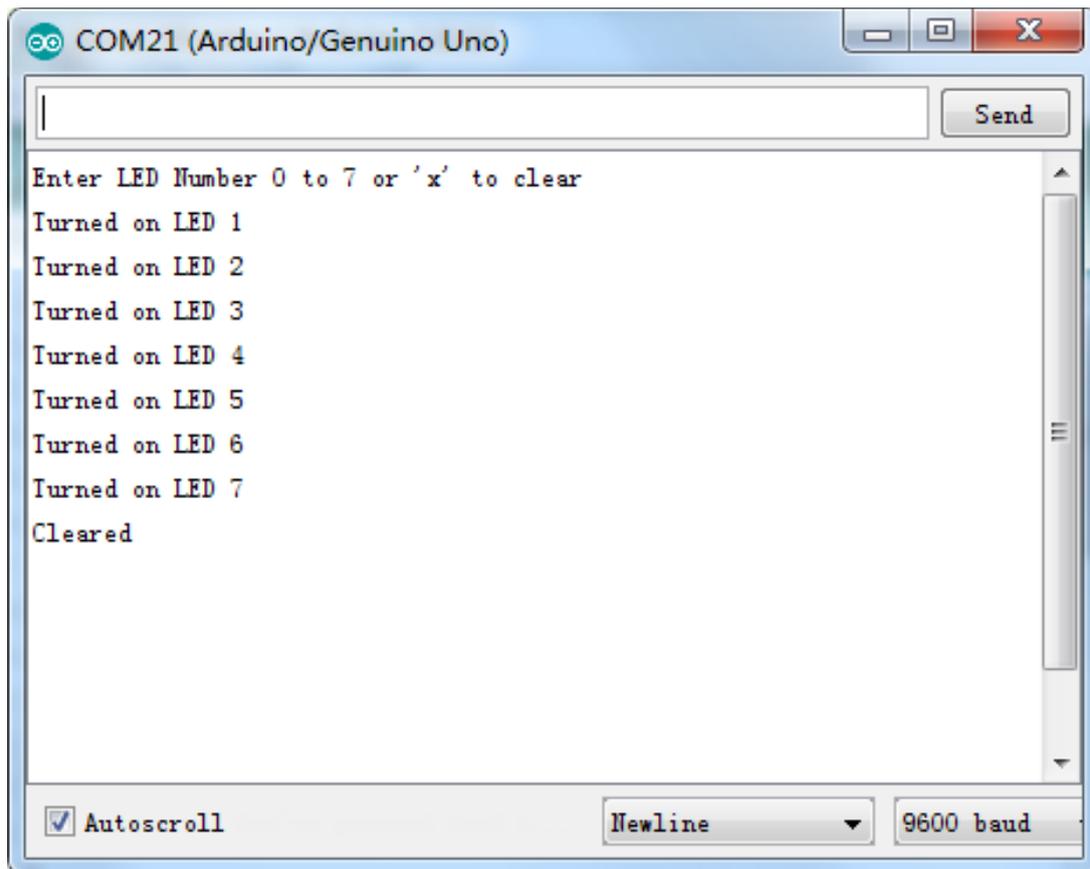


Dieses Fenster ist der Serielle Monitor, der es Ihnen erlaubt sowohl Nachrichten von Ihrem Computer an das UNO Board zu senden, als auch Nachrichten vom UNO Board zu empfangen. Die erste Nachricht, die Sie bekommen, lautet: *„Enter LED Number 0 to 7 or 'x' to clear“*.

Die Nachricht teilt uns mit, welche Befehle wir an das Board senden können: Ein „x“ schaltet alle LEDs aus. Eine beliebige Zahl von 0 – 7 schaltet die jeweilige LED ein, wobei 0 die LED ganz links ist und 7 die LED ganz rechts.

Versuchen Sie die folgenden Befehle nacheinander über das Eingabefeld oben im Seriellen Monitor an Ihr Board zu senden. Drücken Sie nach jedem Befehl *„Senden“*:
x 0 3 5

Der erste Befehl (x) wird nichts auslösen, da die LEDs bereits alle ausgeschaltet sind. Beim Senden der Zahlen sollte jedoch die entsprechende LED aufleuchten und Sie sollten eine Bestätigungsnachricht vom Board im Seriellen Monitor erhalten. Die Anzeige im Seriellen Monitor sollte dann bei Ihnen ähnlich wie hier aussehen:



Geben Sie erneut „x“ ein und klicken Sie auf „Senden“, um alle LEDs auszuschalten.

Code

Nach dem Verbinden der Komponenten öffnen Sie bitte den Sketch im Code-Ordner unter „*Lesson 9 The Serial Monitor*“ und laden ihn auf Ihr UNO Board hoch. Bei Fragen zum Hochladen eines Sketches schauen Sie sich bitte Lektion 2 noch einmal an.

Wie Sie wahrscheinlich bereits geahnt haben, basiert unser Sketch auf dem Sketch aus Lektion 24. Also werden wir hier nur die neuen Teile aufgreifen.

In der setup-Funktion gibt es drei neue Zeilen am Ende.

```
void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  updateShiftRegister();
  Serial.begin(9600);
}
```

```

while (! Serial); // Wait until Serial is ready - Leonardo
Serial.println("Enter LED Number 0 to 7 or 'x' to clear");
}

```

Zuerst haben wir den Befehl „Serial.begin(9600)“. Dieser leitet die Serielle Kommunikation mit einer Baudrate von 9600 ein. Die Baudrate ist die Geschwindigkeit der Datenübertragung. Sie können die Baudrate auf einen höheren Wert stellen, müssen dabei aber auch die Einstellung im Seriellen Monitor anpassen. Zunächst belassen Sie den wert aber bitte bei 9600.

Die Zeile mit der „while“-Schleife wartet solange, bis eine Serielle Verbindung zum Computer hergestellt ist und fährt dann fort.

Ohne diesen Code könnte es passieren, dass die erste Nachricht zu früh gesendet und daher nicht empfangen wird. Dies ist allerdings nur beim Arduino Leonardo notwendig, da beim UNO (ihrem Board) das Board automatisch zurückgesetzt wird, wenn Sie den Seriellen Monitor öffnen.

Die letzte Zeile in der setup-Funktion sendet die „Begrüßungsnachricht“ aus, die wir später oben im Seriellen Monitor sehen werden.

Die ganze Steuerung der LEDs passiert in der loop-Funktion:

```

void loop()
{
  if (Serial.available())
  {
    char ch = Serial.read();
    if (ch >= '0' && ch <= '7')
    {
      int led = ch - '0';
      bitSet(leds, led);
      updateShiftRegister();
      Serial.print("Turned on LED ");
      Serial.println(led);
    }
    if (ch == 'x')
    {
      leds = 0;
      updateShiftRegister();
      Serial.println("Cleared");
    }
  }
}

```

```
    }  
  }  
}
```

Alles, was innerhalb der loop-Funktion passiert, ist in einem if-Statement mit dem Parameter „*Serial.available()*“. Es wird also nur etwas passieren, wenn die Funktion *Serial.available()* „*true*“ (= wahr) zurückgibt.

Serial.available() gibt *true* zurück, wenn Nachrichten über die Serielle Verbindung empfangen wurden. Eingehende Serielle Nachrichten werden in einem sogenannten „*Buffer*“ (= Puffer) gespeichert. Wenn dieser Buffer Nachrichten enthält, gibt *Serial.available()* *true* zurück.

Wenn eine Nachricht empfangen wurde, kann man sie über diesen Befehl lesen:

```
char ch = Serial.read();
```

Die Funktion „*Serial.read()*“ liest das nächste Zeichen aus dem Buffer und entfernt es aus diesem. Hier wird dieses Zeichen gleichzeitig der Variable „*ch*“ zugewiesen. Die *ch*-Variable ist eine Variable des Typs „*char*“, was für Charakter bzw Zeichen steht. Eine *char*-Variable kann also genau ein Zeichen abspeichern.

Wenn Sie sich an die Instruktionen der ersten Nachricht im Seriellen Monitor halten, kann es sich bei dem empfangenden Character um den Buchstaben *x* oder eine Zahl zwischen 0 und 7 handeln.

Das if-Statement in der nächsten Zeile überprüft ob es sich um eine Zahl zwischen 0 und 7 handelt, indem es überprüft ob es sich um eine Zahl handelt, die größer oder gleich 0 und gleichzeitig kleiner oder gleich 7 ist. Da es sich bei einem Zeichen auch um einen Buchstaben handeln kann, wird nicht direkt das Zeichen verglichen, sondern der sogenannte „*ASCII*“-Wert. (American Standard Code for Information Interchange). Jedes Zeichen hat einen einzigartigen vordefinierten ASCII-Wert in Form einer Zahl und diese Zahlen kann man vergleichen und so herausfinden, um welches Zeichen es sich handelt.

Wenn das if-Statement wahr ist, kommen wir zur nächsten Zeile.

```
int led = ch - '0';
```

Nun rechnen wir mit Charaktern! Wir subtrahieren die 0 vom Charakter, den die *ch*-Variable gerade enthält und speichern das Ergebnis in der Variable „*led*“. Wenn über die Serielle Verbindung eine 0 empfangen, wird (weil $0-0 = 0$) eine 0 in der *led*-Variable gespeichert. Bei einer 7, würde $7-0 = 7$ in der Variable gespeichert.

Da wir nun die Nummer der LED wissen, die angeschaltet werden soll, können wir den entsprechenden Bit in der *leds*-Variable setzen und das Schieberegister updaten.

```
bitSet(leds, led);
```

```
updateShiftRegister();
```

Die nächsten zwei Zeilen geben eine Bestätigungsnachricht an den Seriellen Monitor zurück.

```
Serial.print("Turned on LED ");
```

```
Serial.println(led);
```

Die erste Zeile nutzt die Funktion „*Serial.print()*“ statt „*Serial.println()*“. Der unterschied zwischen den beiden Funktionen ist, dass die letztere zusätzlich zur angegebenen Nachricht noch einen Zeilenumbruch ausgibt, also die nächste Ausgabe in einer neuen Zeile anfangen wird. Erstere Funktion dagegen macht keinen Umbruch und man kann danach mit dem nächsten print-Befehl genau an der Stelle weiterschreiben, wo der letzte print-Befehl aufgehört hat. Wir benutzen in der ersten Zeile *Serial.print()*, da wir keinen Umbruch wollen, weil das nur der erste Teil der Nachricht ist. Mit der zweiten Zeile wird die LED-Nummer angehängt und die Nachricht ist fertig, weshalb wir *Serial.println()* nutzen, was am Ende automatisch in die nächste Zeile wechselt.

Die Nummer der LED wird dabei in der *led*-Variable, eine Variable des Typs „*int*“ (Zahlenvariable) gespeichert. Eigentlich benötigt die *Serial.println*-Funktion einen „*String*“ (Text) als Parameter. Das ist aber nicht schlimm, da die Funktion *int*-Variablen automatisch in Text umwandelt.

Nach dem *if*-Statement, dass sich um Zahleneingaben von 0 bis 7 kümmert, folgt das *if*-Statement, dass sich darum kümmert was passiert, wenn ein *x* empfangen wird (*ch = x*):

```
    if (ch == 'x')
    {
        leds = 0;
        updateShiftRegister();
        Serial.println("Cleared");
    }
```

Bei einem „*x*“ werden alle Bits der *leds*-Variable auf 0 gesetzt, das Schieberegister geupdatet und eine Bestätigungsnachricht geschickt, dass alle LEDs ausgeschaltet wurden.

Lektion 10 Fotozelle

Übersicht

In dieser Lektion lernen Sie, wie man die Lichtintensität mit Hilfe einer Fotozelle über einen analogen Ausgang misst. Wir bauen auf Lektion 8 auf und kontrollieren mit Hilfe der Lichtintensität / Helligkeit die Anzahl der LEDs, die aufleuchten sollen. Die Fotozelle wird links am Breadboard platziert.

Benötigte Bauteile:

- (1) x Elegoo UNO R3
- (1) x 830 Punkte Breadboard
- (8) x LEDs
- (8) x 220 Ohm Widerstände
- (1) x 1K Ohm Widerstand
- (1) x 74HC595 IC
- (1) x Fotowiderstand (Fotozelle)
- (16) x M-M Kabel (Männlich zu Männlich DuPont Jumper Kabel)



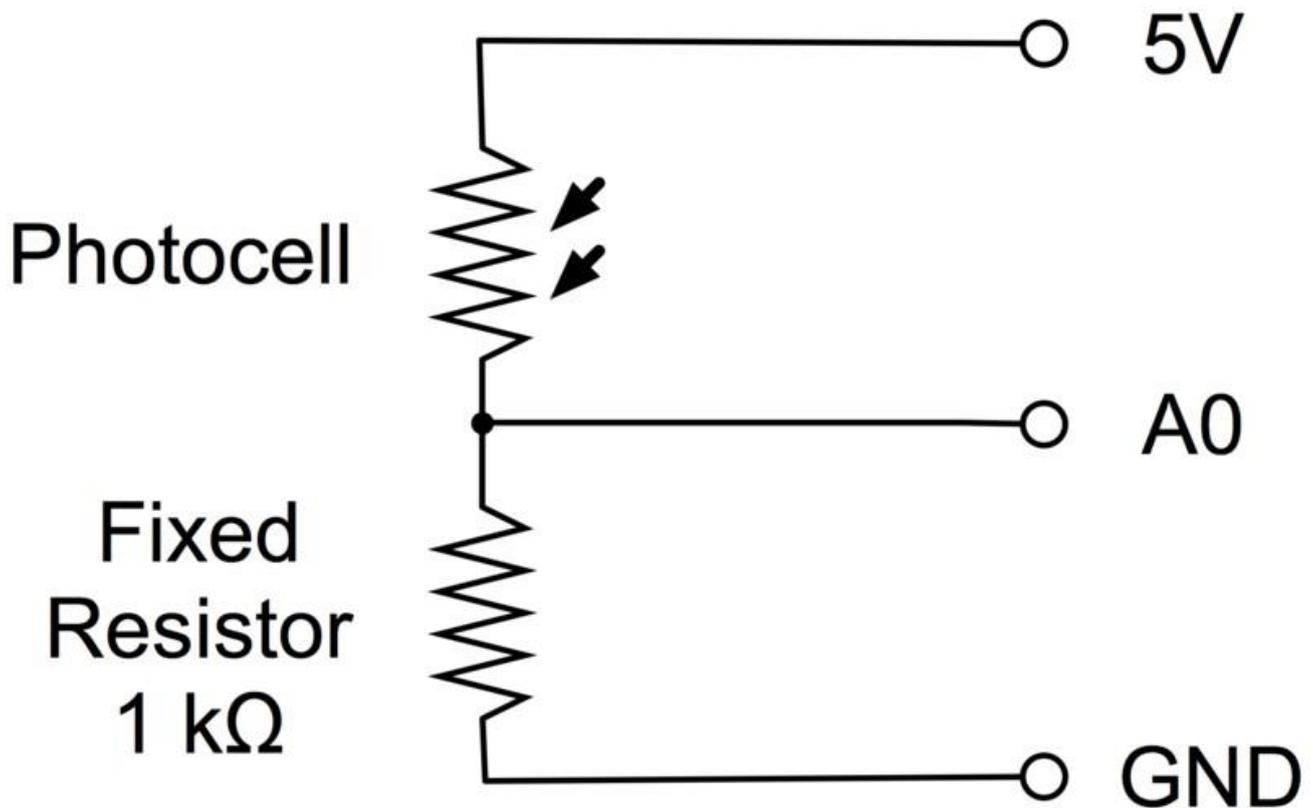
Einführung in die Komponenten

Fotozelle:

Eine Fotozelle ist eine Art lichtabhängiger Widerstand, der manchmal auch „LDR“ genannt wird. Dieses Bauteil arbeitet wie ein normaler Widerstand, mit der Ausnahme, dass der Widerstandswert sich mit der Lichtmenge, die auf ihn strahlt, ändert.

Diese Fotozelle hat in fast kompletter Dunkelheit einen Widerstand von 50 k Ω und in hellem Licht einen Widerstand von etwa 500 Ω . Damit wir den Widerstandswert mit unserem UNO R3 Board über einen analogen Eingang messen können, müssen wir ihn irgendwie in eine Spannung umwandeln.

Der einfachste Weg dies zu tun ist, die Fotozelle mit einem festen Widerstand zu kombinieren.

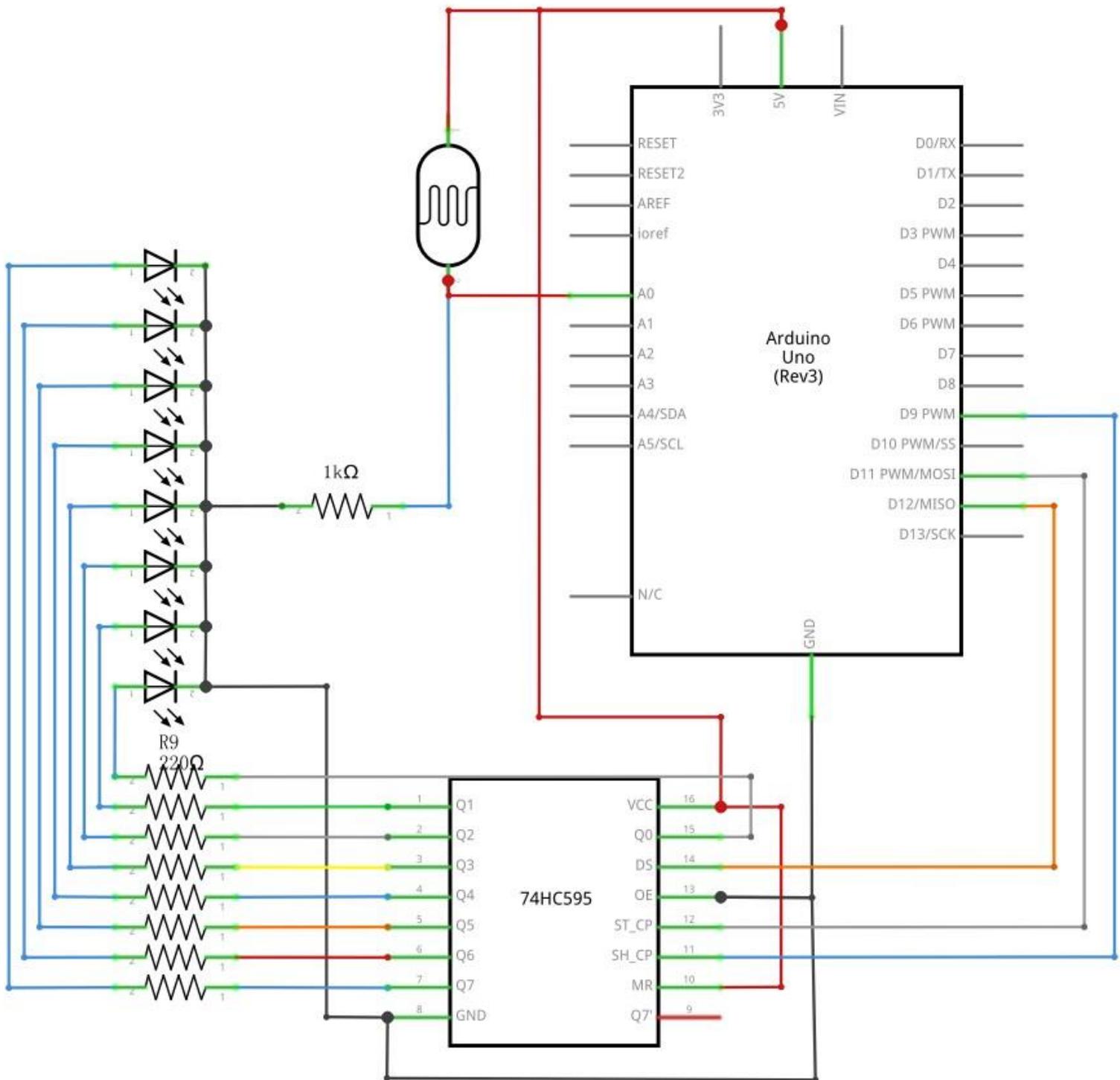


Der feste Widerstand und die Fotozelle zusammen arbeiten ähnlich wie ein Potentiometer. Wenn das Licht hell ist, ist der Widerstand der Fotozelle im Vergleich zum festen Widerstand sehr gering, also verhält es sich wie wenn ein Potentiometer auf das Maximum gestellt ist.

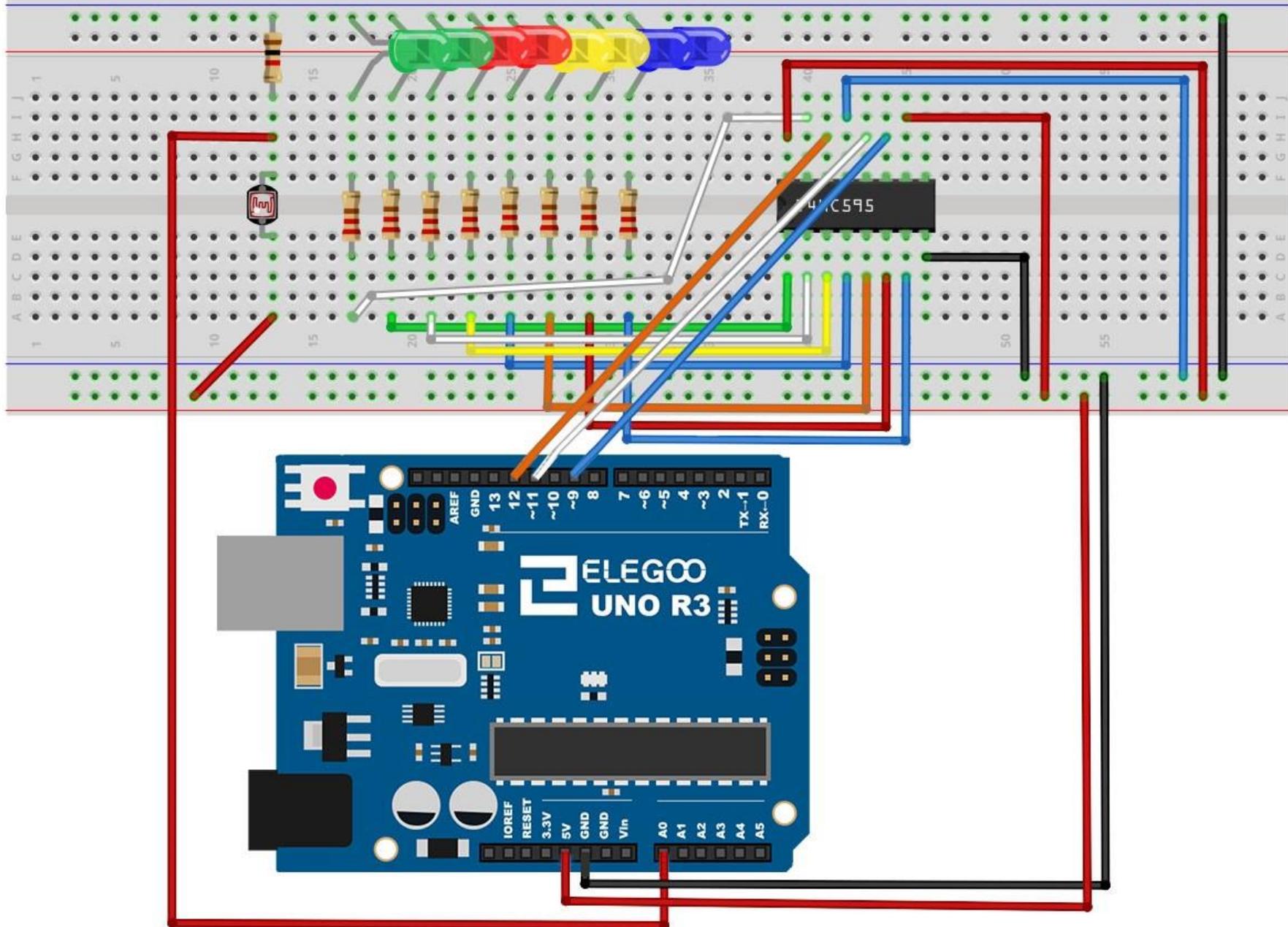
Wenn die Fotozelle sich in dunkler Umgebung befindet, wird ihr Widerstand größer als der des festen 1 kΩ Widerstands und es verhält sich wie wenn ein Potentiometer auf das Minimum (nach GND) gestellt ist.

Laden Sie den Sketch hoch und probieren Sie einmal Ihren Finger auf die Fotozelle zu halten und danach die Fotozelle in die Nähe einer Lichtquelle zu positionieren.

Verbindungsschema



Schaltplan



Code

Nach dem Verbinden der Komponenten öffnen Sie bitte den Sketch im Code-Ordner unter Lektion 10 Photocell und laden ihn auf Ihr UNO Board hoch. Bei Fragen zum Hochladen eines Sketches schauen Sie sich bitte Lektion 2 noch einmal an.

Das erste, was wir beachten müssen, ist, dass der Name des analogen Eingangs „lightPin“ heißt.

Die nachfolgende Zeile berechnet, wie viele LEDs aufleuchten sollen.

```
int numLEDSLit = reading / 57; // all LEDs lit at 1k
```

Wir teilen den Rohwert, den wir am Pin erhalten durch 57. In anderen Worten teilen wir den Widerstandsbereich in neun Zonen auf, von alle LEDs ausgeschaltet bis alle LEDs eingeschaltet. Wenn am Eingang nun ein Widerstand von 1 kΩ anliegt (also die Fozzelle komplett beleuchtet ist), wird der Rohwert etwa $1023 / 2 = 511$ betragen. Bei diesem Wert alle LEDs aufleuchten. Und die „numLEDSLit“-Variable würde $1023 / 57 = 9$ betragen.

Beispielbild

