Teil 2

# Lektion 6

## Passiver Buzzer

# Übersicht

In dieser Lektion lernen Sie einen passiven Buzzer zu benutzen.

Das Ziel dieser Lektion ist es, acht unterschiedliche Töne erklingen zu lassen, jeden Ton für 0,5 Sekunden: Alt Do (523Hz), Re (587Hz), Mi (659Hz), Fa (698Hz), Sol (784Hz), La (880Hz), Ti (988Hz) und das Soprane Do (1047Hz).

## Benötigte Bauteile:

(1) x Elegoo UNO R3

(1) x Passive buzzer

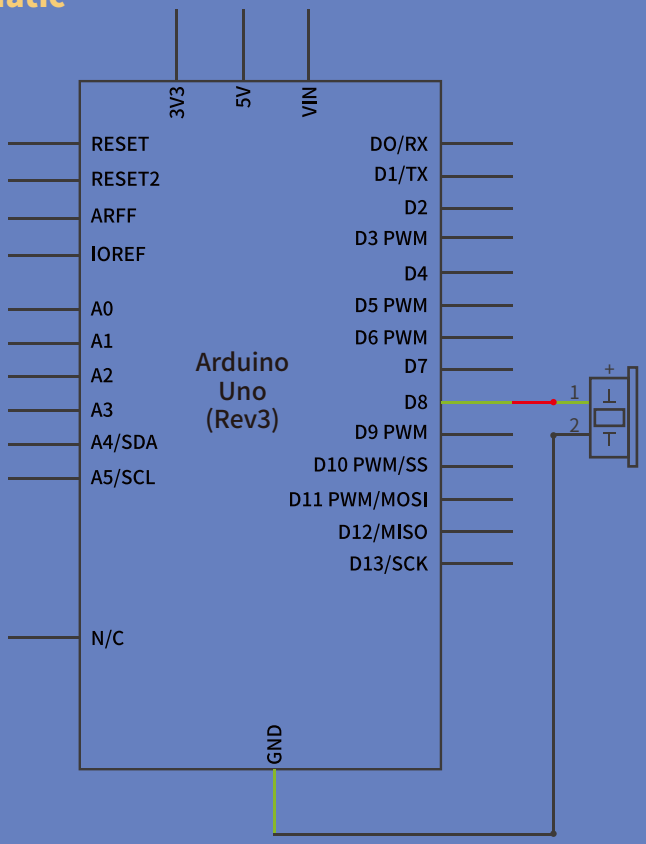
(2) x W-M Kabel (Weiblich zu Männlich DuPont Jumper Kabel)

# Einführung in die Komponenten

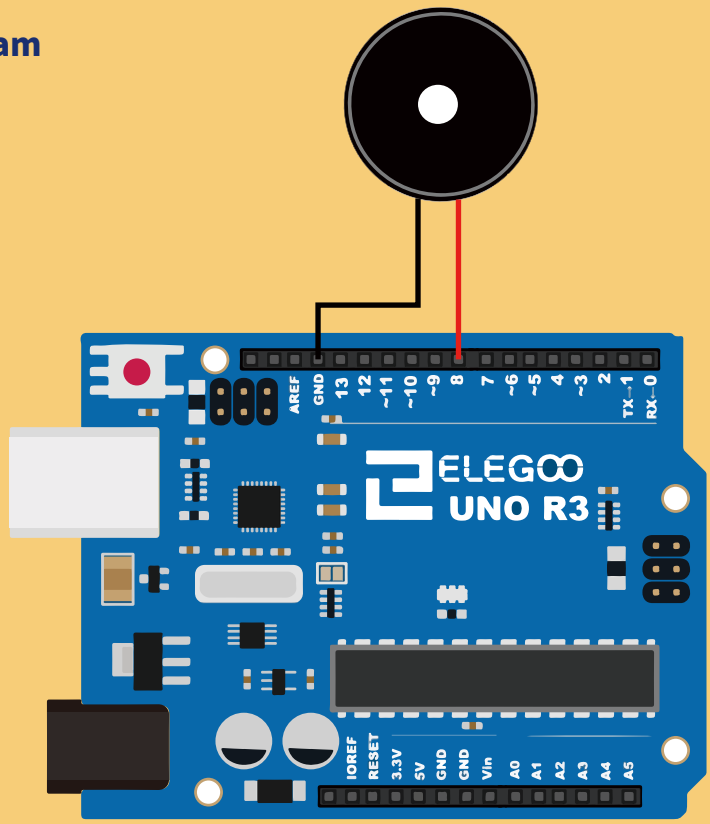
**Passiver Buzzer:**

- **Das** Funktionsprinzip des passiven Buzzers ist es durch PWM-generierte Audiosignale die Luft zum vibrieren zu bringen. Je nachdem wie hoch die Frequenz des Signals ist, werden verschiedene Töne generiert. Ein Audiosignal von beispielsweise 587Hz (Hertz: Frequenzeinheit) wäre ein Alt Do, 587Hz wären einRe und 659Hz wären ein Mi. Mit passiven Buzzern kann man durch schnelle Frequenzwechsel Lieder abspielen.
- **Zum** Generieren des Pulses für das PWM-Signal des passiven Buzzers dürfen wir auf keinen Fall die analogWrite-Funktion des Boards benutzen, da ihr Puls auf 500Hz festgelegt ist und somit ein immergleiches Signal ausgegeben werden würde.

## Connection Schematic



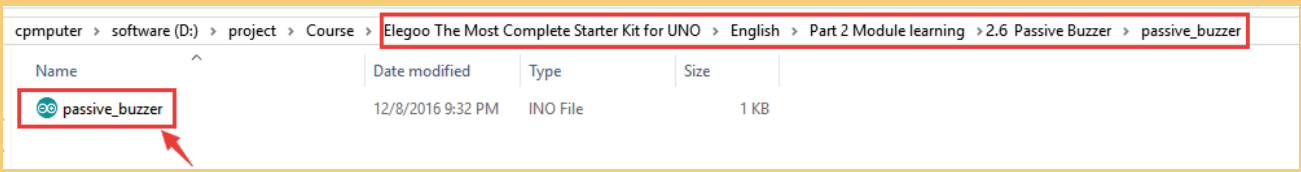

## Wiring diagram




Verbinden Sie den Buzzer mit dem UNO R3 Board. Das rote Kabel muss mit Pin 8 verbunden werden, das schwarze Kabel mit GND.

## Code

- Bitte öffnen Sie das Programm




- Bevor Sie dies ausführen können, stellen Sie sicher, dass Sie die Bibliothek <pitches> installiert haben oder installieren Sie sie gegebenenfalls neu. Andernfalls funktioniert Ihr Code nicht.

- Einzelheiten zum Laden der Bibliotheksdatei finden Sie in Lektion 5 in Teil 1.

```
int melody [] = { NOTE_C5, NOTE_D5, NOTE_E5, NOTE_F5, NOTE_G5, NOTE_A5, NOTE_B5, NOTE_C6};
int melody [] = {....}   : is a array.
```

## array
## [Data Types]
## Description

- Ein Array ist eine Sammlung von Variablen, auf die mit einer Indexnummer zugegriffen wird. Arrays in der Programmiersprache C ++ können kompliziert sein, die Verwendung einfacher Arrays ist jedoch relativ einfach.

### Creating (Declaring) an Array

- Alle folgenden Methoden sind gültige Methoden zum Erstellen (Deklarieren) eines Arrays.
Zum Beispiel:

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
char message[6] = "hello";
```

- Sie können ein Array deklarieren, ohne es (wie in myInts) zu initialisieren.

- In myPins deklarieren wir ein Array, ohne explizit eine Größe auszuwählen. Der Compiler zählt die Elemente und erstellt ein Array mit der entsprechenden Größe.

- Schließlich können Sie Ihr Array wie in mySensVals initialisieren und seine Größe ändern. Beachten Sie, dass beim Deklarieren eines Arrays vom Typ “char” ein Element mehr als Ihre Initialisierung erforderlich ist, um das erforderliche Nullzeichen zu enthalten.

## Zugriff auf ein Array

- Arrays sind mit Null indiziert, das heisst unter Bezugnahme auf die obige Array-Initialisierung befindet sich das erste Element des Arrays auf Index 0.
- mySensVals[0] == 2, mySensVals[1] == 4 und so weiter.
- Dies bedeutet auch, dass in einem Array mit zehn Elementen der Index neun das letzte Element ist.
  Zum Beispiel:

```
int myArray[10]={9, 3, 2, 4, 3, 2, 7, 8, 9, 11};
// myArray[9]   contains 11
// myArray[10]   is invalid and contains random information (other memory address)
```

- Aus diesem Grund sollten Sie beim Zugriff auf Arrays vorsichtig sein. Der Zugriff über das Ende eines Arrays hinaus (unter Verwendung einer Indexnummer, die größer als Ihre deklarierte Arraygröße ist) erfolgt aus dem Speicher, der für andere Zwecke verwendet wird. Das Lesen wird wahrscheinlich nicht viel bewirken, außer ungültige Daten zu liefern. Das Schreiben in ungültige Speicherbereiche ist definitiv eine schlechte Idee und führt zu unglücklichen Ergebnissen wie Abstürzen oder zumindest Programmstörungen.
- Im Gegensatz zu BASIC oder JAVA prüft der C ++ - Compiler nicht, ob der Arrayzugriff innerhalb der vorgegebenen Grenzen der von Ihnen angegebenen Array-Definition liegt.

### So weisen Sie einem Array einen Wert zu:

- mySensVals[0] = 10;

### So rufen Sie einen Wert aus einem Array ab:

- x = mySensVals[4]

```
tone(8 , melody[thisNote] , duration);
```

## tone()

- **[Advanced I/O]**

### Beschreibung

Erzeugt eine Rechteckwelle mit der angegebenen Frequenz (und einem Tastverhältnis von 50%) an einem Pin. Eine Dauer kann angegeben werden, andernfalls wird die Welle bis zu einem Aufruf von noTone() fortgesetzt. Der Pin kann an einen Piezo-Summer oder einen anderen Lautsprecher angeschlossen werden, um Töne abzuspielen.

- Es kann jeweils nur ein Ton erzeugt werden. Wenn ein Ton bereits auf einem anderen Pin abgespielt wird, hat der Aufruf von Ton() keine Auswirkung. Wenn der Ton auf demselben Pin abgespielt wird, stellt der Anruf seine Frequenz ein.
  Die Verwendung der Funktion ton() stört die PWM-Ausgabe an den Pins 3 und 11 (auf anderen Devices als dem Arduino Mega).
- Es ist nicht möglich Töne unter 31 Hz zu erzeugen.

### Syntax

```
tone(pin, frequency)
tone(pin, frequency, duration)
```

### Parameters

pin: Der Arduino-Pin, an dem der Ton erzeugt werden soll.
frequency: Die Frequenz des Tons in Hertz. Zulässiger Datentyp: unsigned int.
duration:  Die Dauer des Tons in Millisekunden (optional). Zulässiger Datentype: unsigned long.

### Returns

nichts

### Notes and Warnings

Wenn Sie unterschiedliche Tonhöhen an mehreren Pins abspielen möchten, müssen Sie noTone() an einem Pin aufrufen, bevor Sie ton() am nächsten Pin aufrufen.