Arduino Programmieren

1. Arduino als Beispiel für einen Microcontroller

Ein Arduino ist ein Microcontroller. Das heißt er ist ein Prozessor, der auch nach außen Funktionen besitzt. Auf ihm kann also ein Programm gespeichert werden, welches ausgeführt wird und dabei Auswirkungen nach außen hin hat. Microcontroller werden in fast jeder Elektronik eingesetzt: Haushaltsgeräte, Unterhaltungselektronik, Kraftfahrzeuge, Handy oder Uhren beispielsweise.

1. Hilfreiche Webseiten

Alle Anleitungen finden sich auch digital auf:

<https://funduino.de/anleitung> (Stand: 25.04.22)

Für Tipps, welche Befehle der Arduino kann:

<https://www.arduino.cc/reference/en/#functions>

1. Aufgaben

Bearbeite die ersten 4 Aufgaben (bis d.) auf jeden Fall in der unten angegebenen Reihenfolge. Danach kannst du die Reihenfolge auch ändern und zuerst das bearbeiten, an dem du mehr Spaß hast.

* 1. Einführungskapitel: Arduino Microcontrolling
  2. Blinkende LED
  3. Wechselblinker
  4. LED per Taster aktivieren
  5. Pulsierende LED
  6. Gleichzeitig Ton- und Lichtsignal
  7. RGB LED
  8. Helligkeitssensor
  9. Drehregler
  10. Verschiedene Töne mit dem Arduino spielen
  11. Das Schieberegister am Arduino

**Mit dem Arduino und einem Lautsprecher Töne erzeugen**

Mit dem Arduino lassen sich auf verschiedene Art und Weise Töne erzeugen. Die einfachste Möglichkeit ist die Tonerzeugung mit einem aktiven Lautsprecher (active Buzzer), der lediglich an die Spannung von 5V angeschlossen wird. Die Tonerzeugung übernimmt eine im Inneren eingebaute Elektronik. Der Nachteil liegt jedoch darin, dass ein „active buzzer“ nur einen einzigen Ton erzeugen kann – Melodien oder Sirenengeräusche sind damit nicht möglich.

**Aktive und passive Lautsprecher: ein kleiner Exkurs**

Mit einem passiven Lautsprecher (passive buzzer) hat man die Möglichkeit, mit Hilfe des Arduino Mikrocontrollers verschiedene Töne, Melodien oder Sirenensignale zu generieren, da im passive buzzer keine Elektronik vorhanden ist, die einen Ton vorgibt.

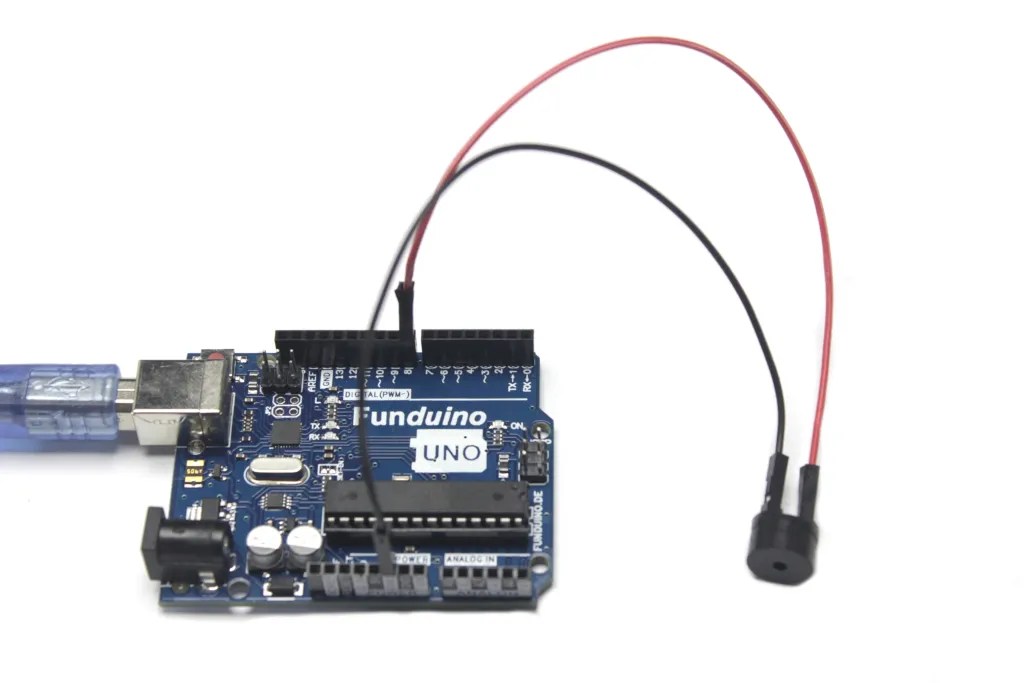
Auf den folgenden Bildern ist zu sehen, wie Piezospeaker und Lautsprecher voneinander zu unterscheiden sind. Links ist jeweils der Lautsprecher dargestellt und rechts der Piezospeaker. AM deutlichsten erkennt man den Lautsprecher an der sichtbaren grünen Platine auf der Unterseite.

Unterseite eines Piezo Speakers

Oberseite eines Piezo Speakers

Aufgabe: Mit einem passiven Lautsprecher sollen unterschiedliche Töne und eine Melodie erzeugt werden.

**Der Schaltplan**



Die Erzeugung des Tones basiert maßgeblich auf dem Befehl **„tone(x, y)“**, wobei der x-Wert den Pin angibt, an dem der Lautsprecher mit der positiven Seite angeschlossen ist und der y-Wert der die Tonhöhe angibt.

Ein Beispiel:

tone(8, 100); // Der Lautsprecher an Pin 8 wird mit der Tonhöhe „100“ aktiviert

delay(1000); // Pause mit 1000 Millisekunden, also 1 Sekunde – Der Lautsprecher bleibt für diese //Zeit aktiv.

noTone(8); // Der Lautsprecher an Pin 8 wird deaktiviert

**Einfache Tonerzeugung**

**Der Programmcode**

void setup() // Im Setup werden keine Informationen benötigt.

//Die Spannungsausgabe für den Piezo-Lautsprecher wird im Sketch durch den Arduino-Befehl "tone" automatisch festgelegt.

{

}

void loop()

{

tone(8, 100); // Im Hauptteil wird nun mit dem Befehl "tone ( x , y )" ein Ton abgegeben.

delay(1000); // mit einer Dauer von 1 Sekunde

noTone(8); // Der Ton wird abgeschaltet

delay(1000); // Der Lautsprecher bleibt eine Sekunde aus

}

**Abwechselnde Tonhöhen erzeugen**

Im zweiten Beispiel werden im Hauptteil (loop) nur ein paar Zeilen ergänzt. Dadurch ertönen zwei Töne abwechselnd mit der Tonhöhe “100” bzw. “200”, wie bei einer Sirene. Eine Pause zwischen den Tönen gibt es nicht.

**Der Programmcode**

void setup()

{

}

void loop()

{

tone(8, 100);

delay(1000);

noTone(8); // An dieser Stelle geht der erste Ton aus.

tone(8, 200); // Der zweite Ton mit der neuen Tonhöhe “200” ertönt.

delay(1000); //… und zwar für eine Sekunde...

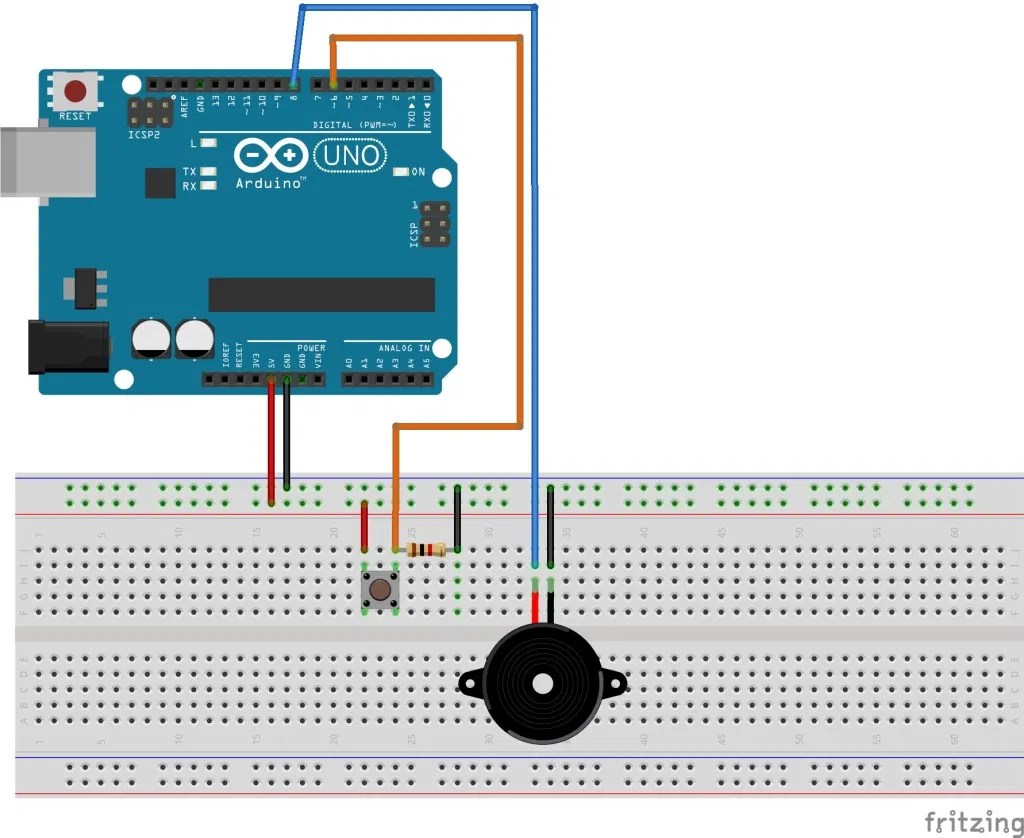
noTone(8); // An dieser Stelle geht der zweite Ton aus und der Loop-Teil des Sketches beginnt von vorn.

}

**Ton erzeugen durch Tastendruck**

Im dritten Beispiel wird ein Taster an Pin6 zum Aufbau hinzugefügt. Im Hauptteil wird durch eine “IF-Abfrage” der Taster ausgelesen. Falls der Taster gedrückt ist, ertönt für eine Sekunde ein Ton mit der Tonhöhe “300”.

**Der Schaltplan**



**Der Programmcode**

int Taster=6;

int Tasterstatus=0;

void setup()

{

pinMode(Taster, INPUT);

}

void loop()

{

Tasterstatus = digitalRead(Taster);

if (Tasterstatus == HIGH) // Wenn der Taster gedrückt ist...

{

tone(8, 300); // …spiele diesen Ton...

delay(1000); //…und zwar für eine Sekunde...

noTone(8); // Ton abschalten

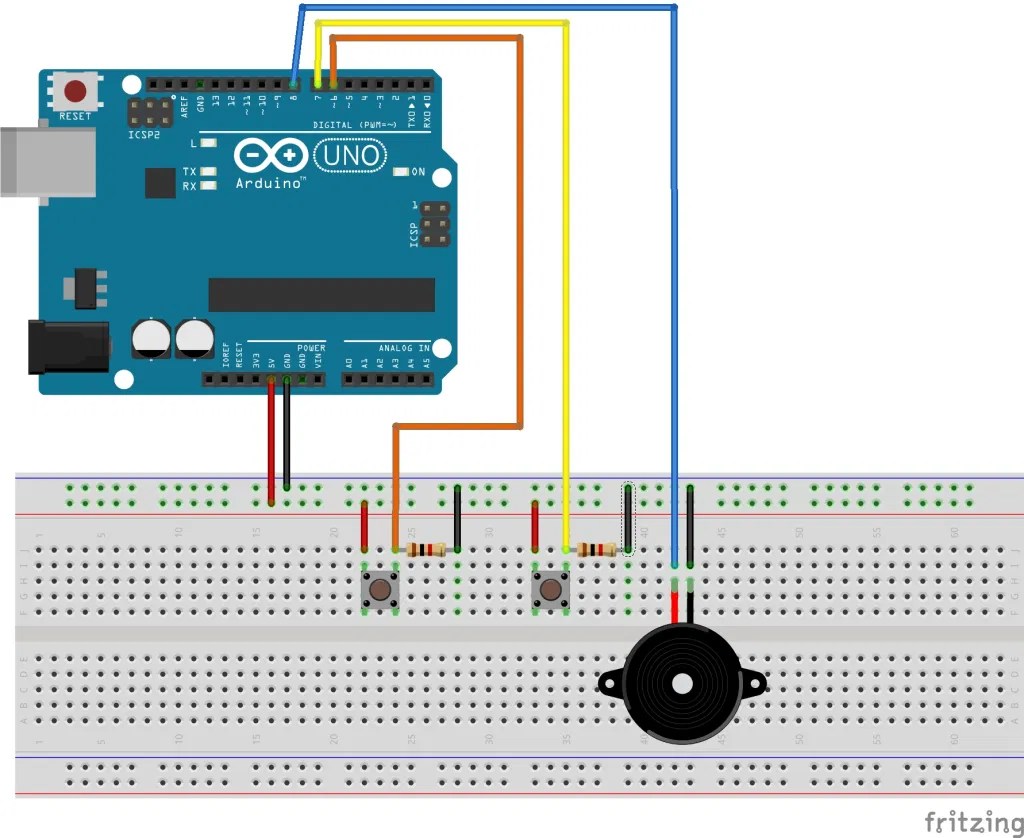
}

}

**Töne in Abhängigkeit von verschiedenen Tasten erzeugen**

Im vierten Beispiel soll je nach Betätigung unterschiedlicher Tasten entschieden werden, welcher Ton vom Lautsprecher abgegeben wird. Dazu wird jeweils ein Taster an den Pins 6 und 7 angeschlossen. Im Hauptteil wird durch zwei “IF-Abfragen” entschieden, welcher Ton abgegeben wird.

**Der Schaltplan**



**Der Programmcode**

int Taster1=6; // Taster1 an Pin6 angeschlossen

int Taster2=7; // Taster2 an Pin7 angeschlossen

int Tasterstatus1=0; // Variable um den Status des Tasters 1 zu speichern.

int Tasterstatus2=0; // Variable um den Status des Tasters 2 zu speichern.

void setup()

{

pinMode(Taster1, INPUT); // Taster1 als Eingang festlegen

pinMode(Taster2, INPUT); // Taster2 als Eingang festlegen

}

void loop()

{

Tasterstatus1 = digitalRead(Taster1); // Status von Taster1 auslesen (HIGH oder LOW)

Tasterstatus2 = digitalRead(Taster2); // Status von Taster2 auslesen (HIGH oder LOW)

if (Tasterstatus1 == HIGH) // Wenn der Taster1 gedrückt ist, dann...

{

tone(8, 100); // Ausgabe eines Tons mit der Tonhöhe 100.

delay (1000); // mit der Dauer von einer Sekunde

noTone(8); // Ausschalten des Tons.

}

if (Tasterstatus2 == HIGH)

{

tone(8, 200);

delay (1000); // mit der Dauer von einer Sekunde

noTone(8); // Ausschalten des Tons.

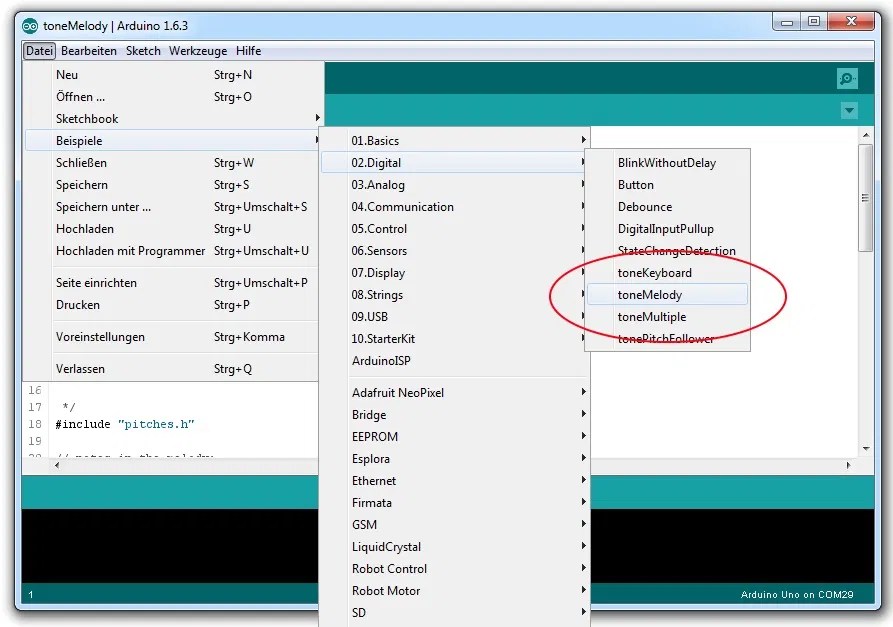
}

}

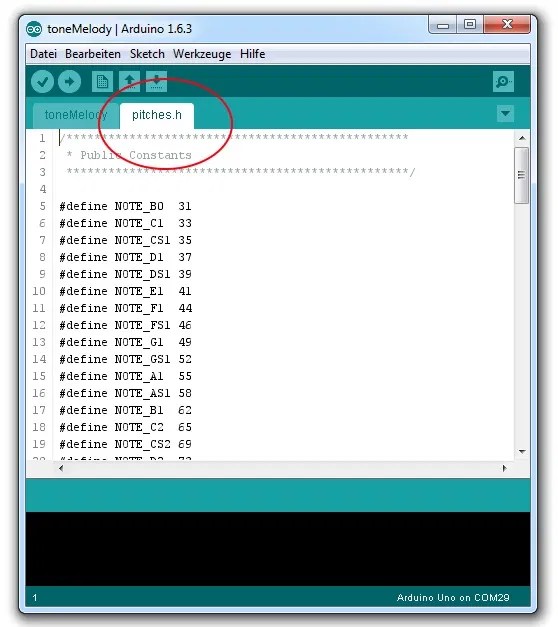
**Melodien mit einem Arduino erzeugen**

In der Arduino-Software ist eine Datei speziell zum Generieren von Tönen mit Hilfe eines Lautsprechers (passive Buzzer) enthalten. Anstelle von Tonhöhen in Form von Zahlen wie bspw. “tone(8, 200)”, können nun auch Töne aus der Tonleiter ausgewählt werden.

Als Beispiel gibt es dafür in der Arduino-Software den Sketch “toneMelody”.

Zu finden unter “Beispiele” -> “02.Digital” -> “toneMelody”

In dem Beispiel ist bereits die Datei hinterlegt, in der sich die Zuordnung von Tonhöhe und Zahlenwert befindet. Sie hat den Namen “pitches.h”. Den Inhalt kann man sich ansehen, wenn man in dem geöffneten Beispielprogramm auf den zweiten Kartenreiter innerhalb des Sketches klickt, wie im folgenden Bild dargestellt.



Um auf die Datei zurückgreifen zu können, muss im Sketch lediglich der Befehl #include „pitches.h“ eingebaut werden und die Datei selber muss als Tab geöffnet sein. Am besten geht das, indem man den Sketch “toneMelody” aus den Beispielen in der Arduino-Software öffnet und dann bearbeitet. Dadurch wird die Datei “pitches.h” automatisch als Tab aufgerufen. Das wird im folgenden Beispiel deutlich.

Dies ist ein ganz kleiner Sketch, der abwechselnd zwei Töne aus der Datei “pitches.h” abspielt.

**Der Programmcode**

#include "pitches.h"

void setup()

{

pinMode (8,OUTPUT); // Lautsprecher an Pin8

}

void loop()

{

tone(8, NOTE\_C4, 1000); // An Pin8 wird die Note C4 für 1000ms gespielt

delay(3000); //Nachdem die Note ertönt, pausiert der Sketch für 3 Sekunden. Das bedeutet, dass nachdem der Ton zu ende gespielt wurde, noch zwei Sekunden Pause ohne Ton verbleiben.

tone(8, NOTE\_G3, 1000); // An Pin8 wird die Note G3 für 1000ms gespielt

delay(3000); //Nachdem die Note ertönt, pausiert der Sketch für 3 Sekunden. Das bedeutet, dass nachdem der Ton zu ende gespielt wurde, noch zwei Sekunden Pause ohne Ton verbleiben.

}

In diesem Sketch wird durch den erweiterten “tone-” Befehl die Beendigung des Tones nicht mehr benötigt.

Befehl: “tone(x, y, z)” x= Pin des Lautsprechers, y= Tonhöhe, z= Dauer des Tons in Millisekunden.

## 1.Schieberegister mit 8 LEDs

Dieser erste Versuch soll einen vordefinierten Zustand im Schieberegister durch 8 LEDs anzeigen.

### Verwendete Bauteile

* [Grundlegende Werkzeuge und Hilfsmittel](https://elektro.turanis.de/html/shop/tools.html)
* [Arduino UNO](https://amzn.to/2HNF665) (oder kompatibles Board)
* [Schieberegister 74HC595](https://amzn.to/2zKTR3q) (hier: SN74HC595N)
* 8x [LED](https://amzn.to/2K9BtYi) (hier: 3mm grün)
* 8x [1kΩ Widerstand](https://amzn.to/2HmrysX)
* [Steckkabel ("Jumperkabel")](https://amzn.to/2wp7K9e)
* [Steckbrett ("Breadboard")](https://amzn.to/2KzQqAB)

### Aufbau

### Aufbau der SchaltungAbb.: Aufbau der Schaltung

### Sketch

#define PIN\_SHIFT 8 // connected to SHCP

#define PIN\_STORE 9 // connected to STCP

#define PIN\_DATA 10 // connected to DS

int ledPattern[8] = {1, 0, 1, 0, 0, 1, 1, 1};

void setup()

{

pinMode(PIN\_STORE, OUTPUT);

pinMode(PIN\_SHIFT, OUTPUT);

pinMode(PIN\_DATA, OUTPUT);

for (int i=0; i<8; i++) {

// set shift pin to "wait"

digitalWrite(PIN\_SHIFT, LOW);

// writing to data pin

digitalWrite(PIN\_DATA, ledPattern[i]);

// rising slope -> shifting data in the register

digitalWrite(PIN\_SHIFT, HIGH);

}

// write whole register to output

digitalWrite(PIN\_STORE, HIGH);

}

void loop ()

{

}

## 2. Binärzähler

Hier werden nacheinander alle 256 Bit-Positionen des Schieberegisters durchgespielt und mit 8 LEDs angezeigt.  
Der Aufbau der Schaltung entspricht der mit "Ex.1"

### Sketch

#define PIN\_SHIFT 8 // connected to SHCP

#define PIN\_STORE 9 // connected to STCP

#define PIN\_DATA 10 // connected to DS

byte count = 0;

// Binary count from 0-255:

// 1 = 000000001

// 2 = 000000010

// 3 = 000000011

// 4 = 000000100

// 5 = 000000101

// etc.

void setup()

{

pinMode(PIN\_STORE, OUTPUT);

pinMode(PIN\_SHIFT, OUTPUT);

pinMode(PIN\_DATA, OUTPUT);

digitalWrite(PIN\_STORE, LOW);

shiftOut(PIN\_DATA, PIN\_SHIFT, LSBFIRST, count);

digitalWrite(PIN\_STORE, HIGH);

delay(2000);

}

void loop ()

{

digitalWrite(PIN\_STORE, LOW);

shiftOut(PIN\_DATA, PIN\_SHIFT, LSBFIRST, count);

digitalWrite(PIN\_STORE, HIGH);

delay(100);

count ++;

}

Quelle: <https://elektro.turanis.de/html/prj061/index.html> [09.05.22]