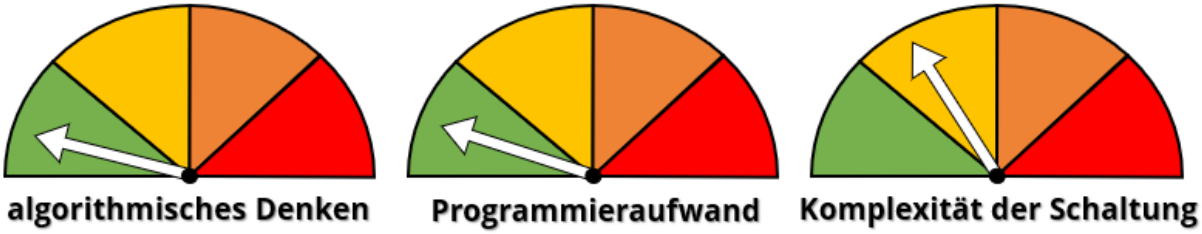
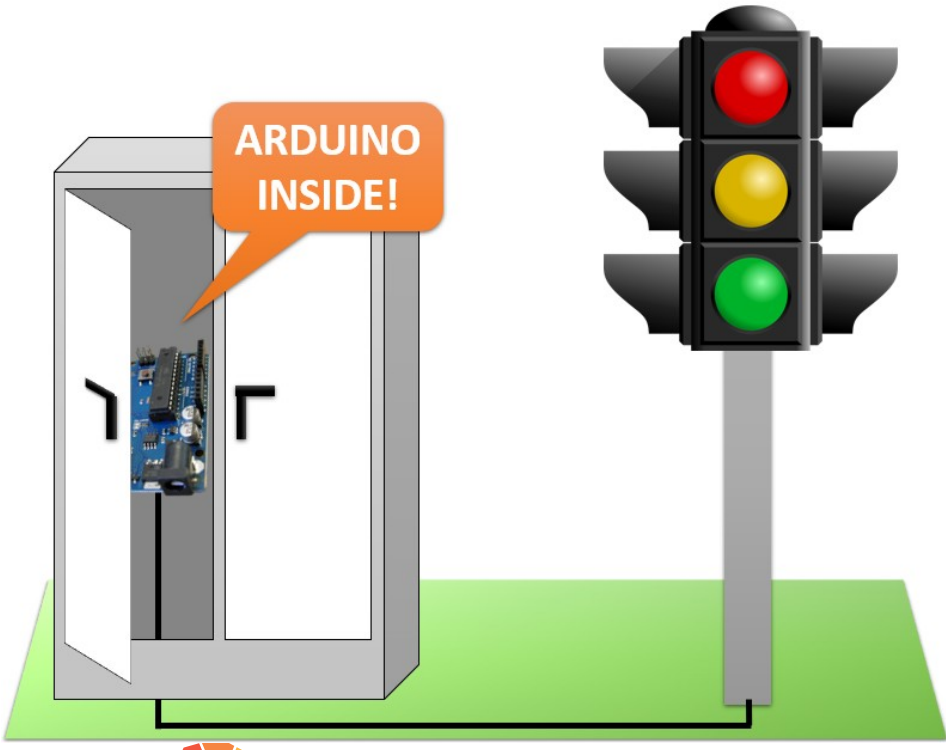
TECHNISCHE UNIVERSITÄT DRESDEN

# Arduino Uno

## Station 5 - ZUSATZ| Halt Stop! – Jetzt fahr ich!




Professur für Didaktik der Informatik

SYLBER Synergetische Lehrerbildung im exzellenten Rahmen

Das Maßnahmenpaket „TUD-Sylber² – Synergetische Lehrerbildung im exzellenten Rahmen“ wird im Rahmen der gemeinsamen „Qualitätsoffensive Lehrerbildung“ von Bund und Ländern aus Mitteln des Bundesministeriums für Bildung und Forschung gefördert.

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

## ZUSATZAUFGABE

Wir haben bereits eine Ampelschaltung, welche den Verkehr und die Fußgänger gleichermaßen berücksichtigt. Aber es gibt viele Menschen, welche auch im Alltag besonders auf Hilfe angewiesen sind. Gerade im Fall der Ampel, ist dir ja bereits schon einmal ein akustischer Warnton aufgefallen. Dieser gibt blinden Menschen, über unterschiedliche Tonhöhen und Abständen der einzelnen Töne an, ob die Ampel gerade rot oder grün ist. Dies gilt es im nachfolgenden Programm zu realisieren.

## DER PIEZO-LAUTSPRECHER

Zu diesem Zweck schauen wir uns kurz den **Piezo-Lautsprecher** an. Dieser wird mit `GND` und einem digitalem Pin verbunden. Zur Hilfe ist auf dem Lautsprecher ein Pluszeichen aufgedruckt, dort wird das Kabel zum digitalen Pin angesteckt.

Welche Befehle benötigen wir zur Steuerung des Lautsprecher? Zum Starten des Lautsprechers, mit einer ausgewählten Tonfrequenz brauchst du:

```
tone(PIN, TONHÖHE);
```

als Befehl. Soll der Ton wieder ausgeschalten werden kannst du dies mit:

```
noTone(PIN);
```

realisieren.

## TONFOLGE FÜR EINE GRÜNE VERKEHRSAMPEL

Solange der Taster nicht gedrückt ist werden die Befehle für die grüne Verkehrsampel und die rote Fußgängerampel ohne Wartezeiten ausgeführt. Demnach können wir unsere Tonfolge recht komfortabel realisieren. Dazu gibst du zuerst einen Ton aus, dann stellst du eine Pause mit Hilfe der `delay()`-Funktion ein und stoppst den Ton wieder. Auch nach dem Stoppen ist wieder eine Wartezeit notwendig, da die anderen Befehle sonst so schnell abgearbeitet werden würden, dass der Ton direkt wieder gestartet wird.

## AUFGABE 1 – TONAUSGABE BEI GRÜNER VERKEHRSAMPEL

Verbinde den Piezo-Lautsprecher zunächst mit dem Arduino, dann initialisierst du dir wieder eine Variable, welche die Pinnummer beinhaltet, an dem der Lautsprecher angeschlossen ist.

Nachfolgend fügst du in der `loop()`-Methode, außerhalb der Verzweigung, die Befehle für die Tonausgabe ein. Teste im Anschluss dein Programm und variiere bei Bedarf Wartezeiten und Tonhöhen.

### TONAUSGABE BEI GRÜNER FUSSGÄNGERAMPEL

Jetzt wird etwas komplizierter. Wir haben ja zwischen den Ampelphasen bereits Pausenzeiten gesetzt. Wie könnte das jetzt an dieser Stelle realisiert werden? Wir gehen mal davon aus, dass wir zwischen zwei Phasen eine Wartezeit von 2000 Millisekunden gesetzt haben. Diese teilen wir jetzt auf und nutzen diese Zeitdauer zur Ausgabe der Tonfolge. Dies realisieren wir in 500 Millisekunden Blöcken und sieht wie folgt aus:

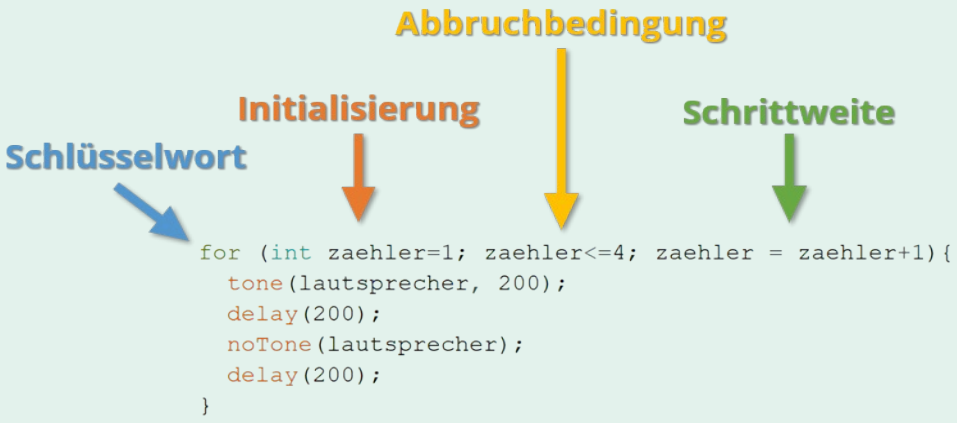
```
tone(PIN, TONHÖHE);
delay(250);
noTone(PIN);
delay(250);
```

Um jetzt unsere ursprünglichen Wartezeiten aufzuaddieren, nutzen wir die sogenannte **Zählschleife**.

## DIE ZÄHLSCHLEIFE (TEIL 1)

Die Zählschleife wiederholt die in ihr befindlichen Befehle so oft, wie es innerhalb der Schleifendefinition vorgegeben ist. Dazu geben wir einen Startwert (Initialisierung), eine Abbruchbedingung und die Schrittweite vor. Solange diese Bedingung erfüllt ist, werden die Befehle ausgeführt und der Zähler verändert. Wie dies beispielhaft im Programmcode aussieht, kannst du auf der nächsten Seite, beim Teil 2, sehen.

## DIE ZÄHLSCHLEIFE (TEIL 2)




Um bei unserem Beispiel zu bleiben, müssen wir die Tonfolge vier mal durchlaufen lassen um auf unsere ursprüngliche Wartezeit von 2000 Millisekunden zu kommen. Dies realisieren wir mit dem folgenden Code.

```
for(int i=1; i<=4, i++){
      tone(PIN, TONHÖHE);
      delay(250);
      noTone(PIN);
      delay(250);
}
```

## AUFGABE 2 - TONAUSGABE BEI GRÜNER FUSSGÄNGERAMPEL

Analysiere zunächst deine Pausenzeiten. Überlege dir eine sinnvolle Zeitspanne für das Abspielen eines Tons, um diesen im Anschluss effektiv aufaddieren zu können.

Implementiere dann die Zählschleifen und ersetze die Wartezeiten zwischen den Ampelphasen mit Tonfolgen. Teste im Anschluss dein Programm und optimiere die jeweiligen Zeiten.


*Grafik auf dem Deckblatt:* *Ampel, Clker-Free-Vector-Images, Pixabay License, https://pixabay.com/de/service/license/, https://pixabay.com/de/vectors/ampel-rot-schwarz-gr%C3%BCn-gelb-24177/*
*Screenshots:* *fritzing electronics made by easy und Arduino IDE 1.8.12 (windows)*
*Alle weiteren Grafiken:* *Patrick Binkert, EduInf@TUD*