Arduino – Wiederholungen effizient programmieren

Beim Programmieren eigener Befehle ist es Dir vermutlich aufgefallen: Häufig ist eine Folge von Befehlen mehrmals hintereinander zu codieren, zum Beispiel so:

```
...
delay(1000);
digitalWrite(pin,HIGH);
delay(1000);
digitalWrite(pin,LOW);
delay(1000);
digitalWrite(pin,HIGH);
delay(1000);
digitalWrite(pin,LOW);
delay(1000);
digitalWrite(pin,HIGH);
delay(1000);
digitalWrite(pin,LOW);
...
```

In diesem Beispiel wird offenbar der **Programmblock**

```
delay(1000);
digitalWrite(pin,HIGH);
delay(1000);
digitalWrite(pin,LOW);
```

dreimal identisch wiederholt – und das verursacht schon einen gewissen „Schreibaufwand“; stell‘ dir bloß vor, diese vier Befehle müssten zwanzigmal oder noch öfter wiederholt werden!

Selbstverständlich gibt es aber in der **Programmiersprache C**, die du zum Programmieren des Arduino-Boards verwendest, Möglichkeiten, solche Wiederholungen mit Hilfe sogenannter **Programmschleifen** zu „automatisieren“.
Du lernst in dieser Aktivität die Variante von Programmschleifen kennen, die nach folgendem Muster zu codieren ist:

```
wiederhole solange(die Bedingung in der Klammer erfüllt ist) {
     Befehl(e) im Programmblock der Schleife (Schleifenblock)
}
```

In der Schreibweise (man sagt auch: In der **Syntax**) der Programmiersprache C sieht das für die als Beispiel präsentierte Wiederholung so aus:

```
int counter = 1;
...
while(counter <= 3){
  delay(1000);
  digitalWrite(pin,HIGH);
  delay(1000);
  digitalWrite(pin,LOW);
  counter = counter + 1;
}
...
```

Das Schlüsselwort **while** ist zu lesen als „wiederhole solange“;

und das dreimalige Wiederholen wird durch „Mitzählen“ erreicht:

Einer Variable, die ganzzahlige Werte speichern kann (hier namens **counter**), wird zunächst der Wert 1 zugewiesen.

Der Schleifenblock wird solange wiederholt, solange der Wert dieser Variablen kleiner oder gleich 3 ist (das ist die **Schleifenbedingung**), und

im Schleifenblock wird der Wert dieser Variablen um Eins erhöht.

Etwas viel auf einmal? Auf der nächsten Seite schauen wir uns das noch einmal ganz genau an…

…blättere doch einfach um!

Mit dem **Mitzählen** – damit der **Schleifenblock** genau dreimal ausgeführt wird – verhält es sich so:

Zuerst wird in der Variablen **counter** der Wert 1 gespeichert.

```
int counter = 1;
...
while(counter <= 3){
  delay(1000);
  digitalWrite(pin,HIGH);
  delay(1000);
  digitalWrite(pin,LOW);
  counter = counter + 1;
}
...
```

Dann wird die **Schleifenbedingung** überprüft. In diesem Beispiel wird die Bedingung, ob der Wert, der in der Variablen **counter** gespeichert ist, kleiner oder gleich 3 ist.

Da der Wert von **counter**, 1, **kleiner** oder gleich 3 ist, wird der **Schleifenblock** **ein erstes Mal** ausgeführt;

dabei wird auch der Wert, der in der Variablen **counter** gespeichert ist, um 1 erhöht. Der Wert der Variablen **counter** ist danach nicht mehr 1 sondern 2.

Nachdem der Schleifenblock vollständig ausgeführt worden ist, wird wieder die **Schleifenbedingung** überprüft.

Da der Wert von **counter**, 2, **kleiner** oder gleich 3 ist, wird der **Schleifenblock** **ein zweites Mal** ausgeführt;

dabei wird auch der Wert, der in der Variablen **counter** gespeichert ist, um 1 erhöht. Der Wert der Variablen **counter** ist danach nicht mehr 2 sondern 3.

Nachdem der Schleifenblock vollständig ausgeführt worden ist, wird wieder die **Schleifenbedingung** überprüft.

Da der Wert von **counter**, 3, kleiner oder **gleich** 3 ist, wird der **Schleifenblock** **ein drittes Mal** ausgeführt;

dabei wird auch der Wert, der in der Variablen **counter** gespeichert ist, um 1 erhöht. Der Wert der Variablen **counter** ist danach nicht mehr 3 sondern 4.

Nachdem der Schleifenblock vollständig ausgeführt worden ist, wird wieder die **Schleifenbedingung** überprüft.

Da der Wert von **counter**, 4, **weder kleiner noch gleich** 3 ist, wird der **Schleifenblock** **nicht mehr** ausgeführt.

Insgesamt wurde aber durch das **Mitzählen** erreicht, dass der Schleifenblock genau dreimal ausgeführt wurde!

Wesentlicher Bestandteil des Mitzählens ist das Weiterzählen jeweils um Eins – auch wenn du zählst, zählst du vermutlich: Eins – Zwei – Drei – Vier – Fünf …; der „Trick“ besteht also tatsächlich darin, immer um Eins weiterzuzählen, und das geschieht im Schleifenblock mit dem Befehl

```
counter = counter + 1;
```

Diesen Befehl wollen wir uns auf der nächsten Seite noch genauer ansehen…

…blättere bitte noch einmal um!

Also:

```
int counter = 1;
...
counter = counter + 1;
```

Mit dem Programmbefehl `int counter = 1;` wird im **Speicher** des Computers ein **Bereich reserviert**, in dem genau eine ganze Zahl Platz hat – deswegen ist in der Programmiersprache C auch das **`int`** so wichtig: Damit wird „dem Computer mitgeteilt“, dass im benötigten Speicherbereich eine integer-Zahl, also eine ganze Zahl, Platz finden muss.

Zudem wird mit dem Programmbefehl `int counter = 1;` vereinbart, dass auf diesen **Speicherbereich** über eine **Variable** namens **`counter`** zugegriffen werden kann, und über die Variable namens **`counter`** auch gleich die Zahl 1 in diesem Speicherbereich „gemerkt“ (gespeichert).

`counter = 1;` …ist nämlich so zu verstehen:

…entweder:

Im **Speicherbereich**, auf den die **Variable** namens **`counter`** zugreift, wird die Zahl 1 gespeichert.

…oder ausführlicher:

Der **`int`-Variable**n namens **`counter`** wird der Wert 1 zugewiesen,
und diese Zahl wird in den **Speicherbereich**, auf den diese **Variable** zugreift, hineingeschrieben.

Eine **Variable** kann aber nicht nur einen **Wert** in den **Speicherbereich**, auf den sie zugreift, **hineinschreiben**, sie kann auch den Wert, der dort gespeichert ist, **lesen**.

Beides passiert bei `counter = counter + 1;` :

Wenn der Variablenname auf der rechten Seite des **Zuweisungsoperator**s **=** verwendet wird,
wird der Wert **aus dem Speicherbereich**, auf den die Variable zugreift, **gelesen**…
…und dieser Wert in der Variablen „gemerkt“.

Dann wird mit `counter + 1;` zu diesem Wert **1 dazugezählt**,

und dieser um 1 erhöhte Wert schließlich mit `counter =` … wieder **in den Speicherbereich**, auf den die Variable zugreift, **hineingeschrieben**.

Insgesamt wurde also die Zahl, die „in der **Variable** namens **`counter`**“ bzw. in dem **Speicherbereich**, auf den die **Variable** namens **`counter`** zugreift, gemerkt (gespeichert) ist, um 1 erhöht.

…und wenn das innerhalb eines Schleifenblocks erfolgt, kann so (mit-) gezählt werden…