Methoden

Benötigtes Material

- Arduino Uno
- USB-Kabel für den Arduino
- LED

Beschreibung

Wir empfehlen hier als Vorwissen das Ansteuern von LEDs (siehe AB **Led ansteuern**). Wenn wir größere Programme schreiben, tauchen manche Codeblöcke mehrfach auf. Manchmal unterscheidet sich der Code auch nur ganz wenig, erscheint aber dennoch redundant. Wir wollen nun lernen, wie wir diesen Code zusammenfassen können, um ihn les- und bearbeitbarer zu machen und sogenannten Spaghetti-Code zu vermeiden.

Hier ein Beispiel wie es mit und wie ohne Methoden aussieht:

Ohne Methode:

```
void loop() {

  digitalWrite(mlv,HIGH);
  digitalWrite(mlr, LOW);
  digitalWrite(mrv,HIGH);
  digitalWrite(mrr, LOW);

  delay(5000);

}
```

Mit Methode:

```
void loop() {

  vorwaerts();



delay(5000);

}
```

In dem obigen Beispiel wurde also eine Methode `vorwaerts()` definiert, welche die Steuerung der Motoren übernimmt. Wir können dann direkt ablesen, was im Code passiert.

Neue Methoden werden außerhalb der `setup`- und `loop`-Blöcke wie folgt definiert:

```
void vorwaers(){
    digitalWrite(mlv, HIGH);
    digitalWrite(mlr, LOW);
    digitalWrite(mrv, LOW);
    digitalWrite(mrr, HIGH);
}
```

Nach dem Definieren von Methoden können diese, wie in Bild1 im `loop`-Block aufgerufen werden. Dann werden ihre Rümpfe, also der Bereich in den geschweiften Klammern, ausgeführt. Methoden sind noch zu viel mehr in der Lage, denn wir können ihnen auch Informationen durch sogenannten `Parameter` übergeben, welche von ihnen verarbeitet werden und uns neue Informationen zurückgeben. Die Rückgabe wird als `return` bezeichnet. Wenn wir eine Methode definieren, können wir in den runden Klammern, neben dem Namen der Methode die `Parameter` aufzählen, bei mehreren werden sie mit einem „,“ getrennt aufgelistet. Sie agieren quasi wie Platzhalter, die erst später gefüllt werden. Wenn wir die Methode dann später in der `loop` aufrufen, so setzen wir tatsächliche Werte in diesen `Parameter` ein.

So könnten wir zum Beispiel folgende Methode definieren, die die Motoren für eine gewisse Zeit ununterbrochen steuert. In diesem Fall soll das Roboterfahrzeug sich eine bestimmte Zeit drehen.

```
void drehen(int time){
    digitalWrite(mlv, HIGH);
    digitalWrite(mlr, LOW);
    digitalWrite(mrv, LOW);
    digitalWrite(mrr, HIGH);

    delay(time);
}
```

Im `loop`-Block können wir dann einfach schreiben:

```
void loop(){

    drehen(5000);
}
```

Wenn wir einen `Parameter` angeben, so müssen wir erstmal mitteilen von welchem `Datentyp` dieser ist. Nutze dazu zunächst diese Tabelle:

Datentypen

Datentyp	Beschreibung
`int`	Ganze Zahlen im Bereich -32.768 bis 32.767 und `HIGH` (1) und `LOW` (0)
`float`	Kommazahlen in einem sehr großen Bereich Diese werden mit einem Punkt geschrieben: `4.5`, `-33.557`, `1.0` (**Vorsicht:** Rundungsfehler möglich!)
`char`	Einzelne Zeichen `'a'`, `'k'`, `'+'` (einfaches Anführungszeichen)
`char*`	Zeichenketten: `"hallo"`, `"test 1"`, `"a"` (doppeltes Anführungszeichen)
`boolean`	Wahrheitswerte: `true` (wahr) und `false` (falsch)

Unsere Methoden können auch Dinge berechnen und diese als Ergebnisse zurückgeben. Wie bereits erwähnt nutzen wir dazu das Schlüsselwort `return`. Was auch immer nach dem `return` steht, wird als Ergebnis dieser Methode ausgegeben, wenn wir sie aufrufen. Wir müssen dann nur, statt `void` den Rückgabetyp der Methode schreiben. Nutze dazu die Tabelle mit den Datentypen.

Tipp: Kommentare für Methoden

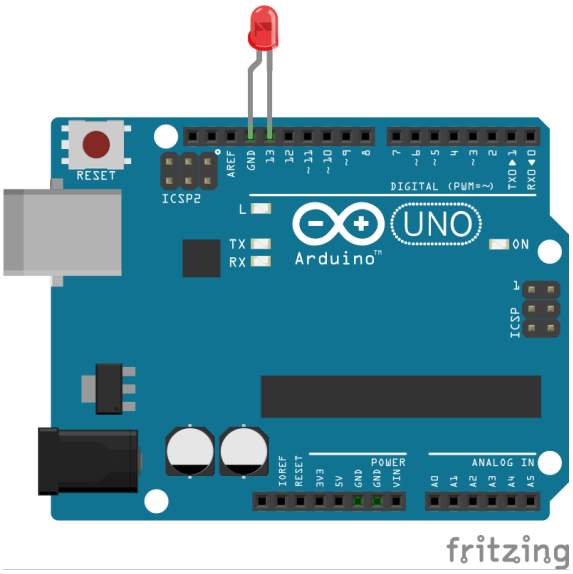
Schreibe über jede deiner Methoden einen Kommentar, der folgende Informationen enthält:

- Nutzen/Verwendungszweck der Methoden.
- Für jeden Parameter: Sinn des Parameters und bei Bedarf nenne Einschränkungen (z.B. wenn nur positive Zahlen erlaubt sind).
- Wenn es einen Rückgabewert gibt: Was wird zurückgegeben?
- Wenn Fehler auftreten können, beschreibe wie und wann…

Mache diese Kommentierung auch in Zukunft **IMMER**. Das nimmt zwar etwas Zeit in Anspruch, lohnt sich aber für größere Projekte sehr und so macht das jede/r gute Programmiererin und Programmierer.

Verkabelung

LED	Arduino	Sonstiges
Anode	Pin13	direkt über Arduino
Kathode	GND	direkt über Arduino



Schaltplan

Im Folgenden wird ein SOS gemorst, wobei wir die Morsezeichen in Methoden ausgelagert haben.

Code

```
//  Der Pin Nr. 13 heißt von nun an LED.
int LED = 13;

void setup() {
  // Der Pin LED soll ein Signal erhalten können (hier Strom).
  // Er fungiert als Akteur.
  pinMode(LED, OUTPUT);
}

void loop() {
  // Morse ein SOS.
  // Rufe dazu die Methoden S und O auf, und übergebe
  // die Zeit, wie lange die LED leuchten soll.
  s(300);
  delay(300);  // kurze Verzögerung, damit die Blinker nicht verschmelzen
  o(1000);
  s(300);

  // Das Programm und das System verweilen in ihrem aktuellen Zustand für
  // 1000ms.
  delay(2000);
}

// Methode um ein S zu morsen.
// sie gibt keinen Wert zurück, deswegen ist sie void.
// Ein Parameter, welcher die Zeit bestimmt wird hier mit übergeben.
void s(int time) {
  // Der Pin LED erhält das Signal, dass dort Strom fließen soll.
  digitalWrite(LED, HIGH);
  // Das Programm und das System verweilen in ihrem aktuellen Zustand für
  // eine Zeit bestimmt durch die Variable time.
  delay(time);
```

```
  // Der Pin LED erhält das Signal, dass dort kein Strom fließen soll.
  digitalWrite(LED, LOW);
  // Das Programm und das System verweilen in ihrem aktuellen Zustand für
  // eine Zeit bestimmt durch die Variable time.
  delay(time);

  digitalWrite(LED, HIGH);
  delay(time);
  digitalWrite(LED, LOW);
  delay(time);

  digitalWrite(LED, HIGH);
  delay(time);
  digitalWrite(LED, LOW);
  delay(time);
}

// Methode um ein O zu morsen.
// Sie gibt keinen Wert zurück, deswegen ist sie void.
// Ein Parameter, welcher die Zeit bestimmt wird hier mit übergeben.
void o(int time) {
  // Der Pin LED erhält das Signal, dass dort Strom fließen soll.
  digitalWrite(LED, HIGH);
  // Das Programm und das System verweilen in ihrem aktuellen Zustand für
  // eine Zeit bestimmt durch die Variable time.
  delay(time);
  // Der Pin LED erhält das Signal, dass dort kein Strom fließen soll.
  digitalWrite(LED, LOW);
  // Das Programm und das System verweilen in ihrem aktuellen Zustand für
  // eine Zeit bestimmt durch die Variable time.
  delay(time);

  digitalWrite(LED, HIGH);
  delay(time);
  digitalWrite(LED, LOW);
  delay(time);

  digitalWrite(LED, HIGH);
  delay(time);
  digitalWrite(LED, LOW);
  delay(time);
}
```

Aufgaben

1. **Erkläre anhand einiger Methoden in diesem Kapitel, was ist der Name der Methoden, Rückgabewert, eine Parameterliste, der Rumpf?**
2. **Prüfe in deinem bisherigen Code, an welchen Stellen du sinnvoll Methoden verwenden kannst und setze diese um. Schreibe Kommentare!**
3. **Kann man Methoden in Methoden aufrufen? Wann ja, hast du dafür ein sonnvolles Beispiel, z.B. in deinem Code?**

Ja es ist möglich.

4. **Methoden können Parameter und Rückgabewerte haben. Überlege dir ein sinnvolles Beispiel, wo man dies einsetzen kann, z.B. in deinem Code!**
5. **Was bedeutet `void` als Rückgabewert einer Methode?**

6. **Was sind `loop` und `setup`?**

7. **Probiert es selbst mal aus und morst ein paar Buchstaben oder sogar Nachrichten an eure Sitznachbarin oder Sitznachbarn.**