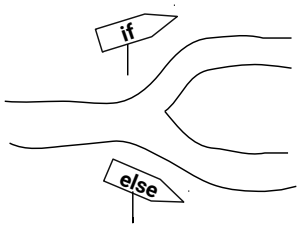
Arbeitsblatt 5 – if-else-Anweisung

Eine if-else-Anweisung ist eine Verzweigung:



Die Kontrollstruktur **if-else** ermöglicht, dass in Abhängigkeit von einer **Bedingung** bestimmte Anweisungen ausgeführt werden und andere dagegen nicht! **Falls (if)** die Bedingung **wahr** ist, dann werden die ***Anweisungen1*** ausgeführt, **sonst (else)**, also wenn die Bedingung **falsch** ist, werden die ***Anweisungen2*** ausgeführt:

```
if (Bedingung) {
  Anweisungen1;
} else {
  Anweisungen2;
}
```

Wir schauen uns mal ein konkretes Beispiel an:

```
4  if (2 == 2) {
5    bob3.setEyes(WHITE, WHITE);
6  }
7  else {
8    bob3.setEyes(OFF, OFF);
9  }
```

Wenn man dieses Programm auf dem BOB3 laufen lässt, dann bekommt man immer dasselbe Ergebnis: **die Augen leuchten weiß**!

Begründung: Da die **Bedingung** „`2 == 2`“ **wahr** ist, wird der if-Zweig, also in diesem Beispiel die Anweisung „`bob3.setEyes(WHITE, WHITE);`“ ausgeführt.

Wir schauen uns ein anderes Beispiel an:

```
4   if (2 == 200) {
5     bob3.setEyes(WHITE, WHITE);
6   }
7   else {
8     bob3.setEyes(OFF, OFF);
9   }
```

Wenn man dieses Programm auf dem BOB3 laufen lässt, dann bekommt man immer dasselbe Ergebnis: **die Augen sind aus**!

Begründung: Da die **Bedingung** „`2 == 200`“ **falsch** ist, wird der else-Zweig, also in diesem Beispiel die Anweisung „`bob3.setEyes(OFF, OFF);`“ ausgeführt.

Eine Bedingung nennt man auch Wahrheitswert oder auch booleschen Ausdruck:

Eine Bedingung ist entweder erfüllt, also **wahr** oder nicht erfüllt, also **falsch**. Der Ausdruck „2 == 2“ ist wahr und der Ausdruck „2 == 200“ ist falsch. Den Operator „==“ nennt man **Vergleichsoperator**.

Vergleichsoperatoren:

Mathe	Vergleichsoperator	Beispiel	Erklärung
=	==	a == b	→ ergibt „**wahr**“, falls die Werte a und b **gleich** sind
≠	!=	a != b	→ ergibt „**wahr**“, falls die Werte a und b **ungleich** sind
>	>	a > b	→ ergibt „**wahr**“, falls a **größer** als b ist
<	<	a < b	→ ergibt „**wahr**“, falls a **kleiner** als b ist
≥	>=	a >= b	→ ergibt „**wahr**“, falls a **größer oder gleich** b ist
≤	<=	a <= b	→ ergibt „**wahr**“, falls a **kleiner oder gleich** b ist

Programmierer verwenden für „***wahr***“ das Wort „***true***“ und für „***falsch***“ das Wort „***false***“.

Aufgabe 1: Was macht der BOB3, wenn du das folgende Programm überträgst?

```
4   if (1000 == 1000) {
5     bob3.setEyes(WHITE, WHITE);
6   }
7   else {
8     bob3.setEyes(OFF, OFF);
9   }
```

Auge 1: Auge 2:

Aufgabe 2: Was macht der BOB3, wenn du das folgende Programm überträgst?

```
4   if (3 == 8) {
5     bob3.setEyes(OFF, OFF);
6   }
7   else {
8     bob3.setEyes(ORANGE, WHITE);
9   }
```

Auge 1: Auge 2:

Aufgabe 3: Was macht der BOB3, wenn du das folgende Programm überträgst?

```
4   if (3 != 3) {
5     bob3.setEyes(OFF, OFF);
6   }
7   else {
8     bob3.setEyes(WHITE, WHITE);
9   }
```

Auge 1: Auge 2:

Aufgabe 4: Was macht der BOB3, wenn du das folgende Programm überträgst?

```
4   if (3 != 8) {
5     bob3.setEyes(OFF, OFF);
6   }
7   else {
8     bob3.setEyes(WHITE, WHITE);
9   }
```

Auge 1: Auge 2:

Aufgabe 5: Was macht der BOB3, wenn du das folgende Programm überträgst?

```
4  if (3 < 8) {
5    bob3.setEyes(SEAGREEN, PURPLE);
6  }
7  else {
8    bob3.setEyes(ORANGE, WHITE);
9  }
```

Auge 1: Auge 2:

Aufgabe 6: Was macht der BOB3, wenn du das folgende Programm überträgst?

```
4  if (8 <= 8) {
5    bob3.setEyes(SEAGREEN, PURPLE);
6  }
7  else {
8    bob3.setEyes(ORANGE, WHITE);
9  }
```

Auge 1: Auge 2:

Aufgabe 7: Was macht der BOB3, wenn du das folgende Programm überträgst?

```
4  if (1001 < 1000) {
5    bob3.setEyes(RED, OFF);
6  }
7  else {
8    bob3.setEyes(OFF, RED);
9  }
```

Auge 1: Auge 2:

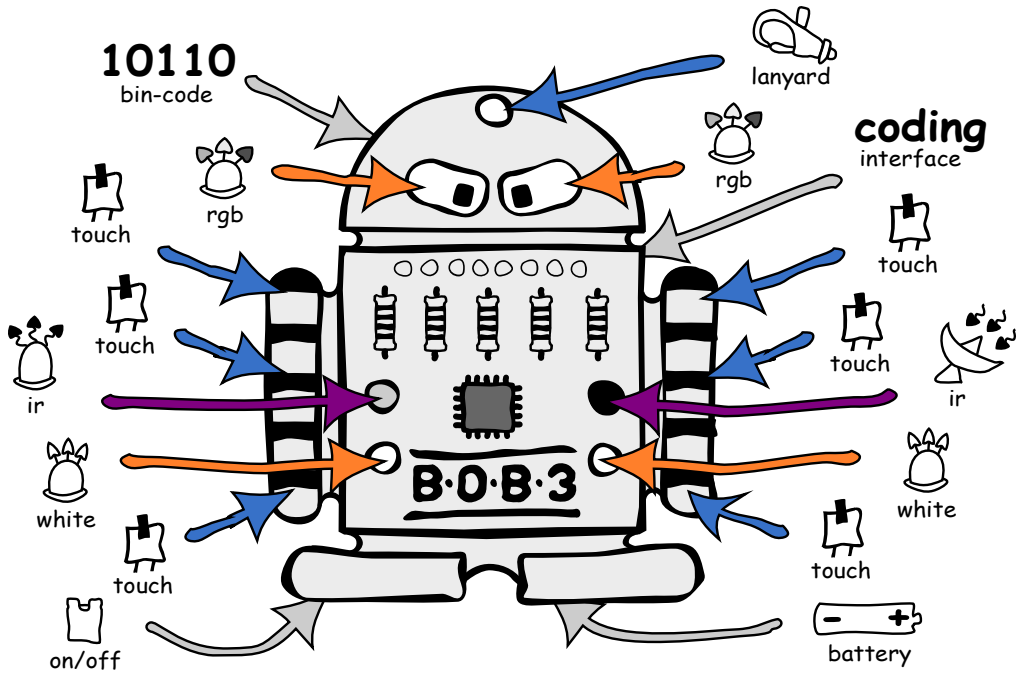
Aufgabe 8: Was macht der BOB3, wenn du das folgende Programm überträgst?

```
4  if (1001 >= 1000) {
5    bob3.setEyes(RED, OFF);
6  }
7  else {
8    bob3.setEyes(OFF, RED);
9  }
```

Auge 1: Auge 2:

Arbeitsblatt 6 – Bob's Hardware

Aus welchen Bestandteilen besteht der BOB3?



BOB3 ist ein kleiner Roboter der merkt, ob seine Arme berührt werden und wenn ja, wo! Er kann Freunde erkennen, seine weißen Scheinwerfer einschalten, seine Augen in allen Farben blinken lassen, nah und fern unterscheiden und einiges mehr! Du kannst ihn frei programmieren, ihm einen eigenen binären Code geben oder ihn mit einer Knopfzelle und dem Lanyard als blinkendes Gadget um den Hals tragen!

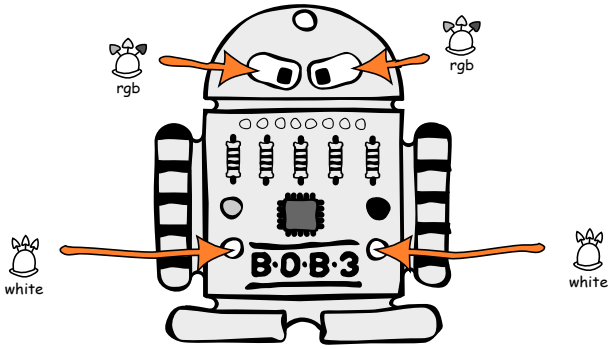
Mit welchen Bauteilen und wie genau macht der Bob das alles?

In diesem Arbeitsblatt schauen wir uns nacheinander die Bestandteile des BOB3 an und erklären, wie diese funktionieren.

Fangen wir mal mit den LEDs, also mit den Leuchtdioden an:

1) LEDs - Leuchtdioden:

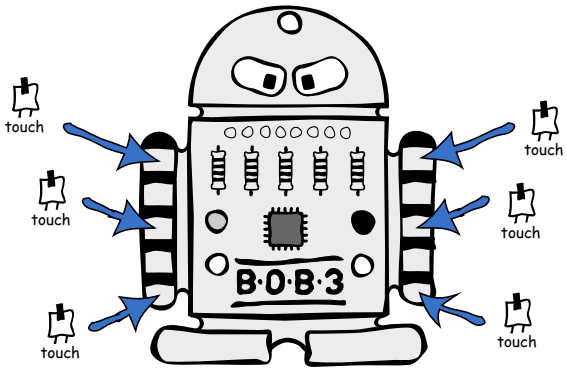
Der Bob hat zwei **RGB-LEDs** als Augen. LEDs sind Lampen, die du ein- und ausschalten kannst. RGB-LEDs können in **allen Farben** leuchten: rot, grün, blau, gelb, orange, hellgrün, dunkelgrün, lila, violett und in ganz vielen anderen Farben!

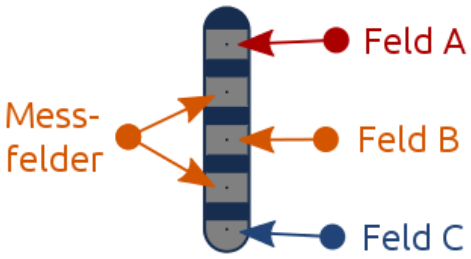


Am Bauch hat der Bob noch zwei superhelle **weiße LEDs** als Scheinwerfer, die sehr hell leuchten können. Damit kannst du den Bob z.B. als Taschenlampe verwenden!

2) Multifeld-Touch-Sensoren:

Beide Arme vom BOB3 sind **Touch-Sensoren**. Die Arme „merken“ also, ob sie berührt werden, oder nicht! Weil der Bob sogar bemerkt, **wo** du den jeweiligen Arm berührst (oben, mitte, unten) sind es **Multifeld-Touch-Sensoren**. Der Bob hat also insgesamt sechs Tastsensoren, die du ansteuern oder abfragen kannst!



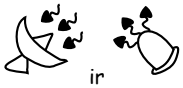


Jeder Arm hat 5 Felder:

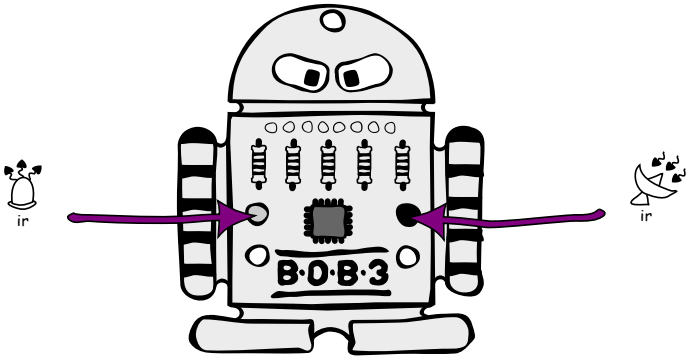
3 Aktivierungsfelder (A, B und C) und
2 Messfelder.

Sobald du ein Aktivierungsfeld **gleichzeitig** mit einem Messfeld berührst, bekommt der Bob ein Signal, ob Feld A, Feld B oder Feld C berührt wurde.

3) IR-Sensor:

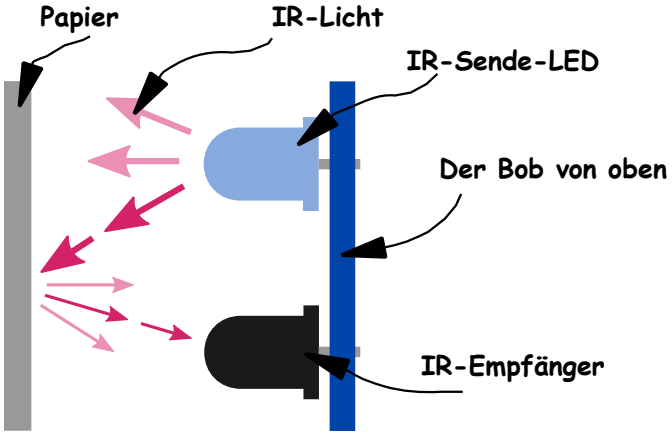


Der BOB3 hat einen **IR-Sensor**, der aus zwei Teilen besteht: Einer violetten **IR-Sende-LED** und einem schwarzen **IR-Empfänger**. Die Abkürzung „IR“ steht für „Infrarot“. Infrarotlicht ist eine spezielle Lichtart.



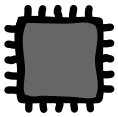
Die IR-Sende-LED **sendet Infrarotlicht** aus und der IR-Empfänger **detektiert das Infrarotlicht**. Mit diesem Sensor kann der Bob nah und fern unterscheiden, bemerken, ob z.B. deine Hand oder ein Blatt Papier vor ihm ist oder er kann anderen BOB3-Robotern Botschaften senden!

Die **Detektion von nah oder fern** funktioniert nach dem **Reflektionsverfahren**: Die IR-Sende-LED sendet IR-Licht aus, dieses trifft dann auf ein Hindernis (z.B. ein Blatt Papier), wird von dem Hindernis **zurückreflektiert** und kann so von dem IR-Empfänger empfangen werden.

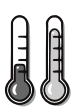


Je näher das Papier vor dem Sensor ist, **desto mehr IR-Licht detektiert** der IR-Empfänger. So kannst du den BOB3 z.B. als Lichtschranke programmieren!

4) Mikrocontroller:



Als „Gehirn“ hat der BOB3 einen **Mikrocontroller**, der den Programmcode ausführt. Der Controller ist mit der gesamten Elektronik vom Bob verbunden und ist die zentrale Rechen- und Steuereinheit des Roboters.



Der Controller vom BOB3 hat sogar einen **Temperatursensor** integriert. Damit kann der Bob **warm** und **kalt** unterscheiden!

Aufgabe 1: Wie viele **Leuchtdioden** hat der BOB3, die **weiß** leuchten können?
Kreuze die richtige Antwort an:

- ☐ fünf
- ☐ eine
- ☐ zwei
- ☐ drei
- ☐ vier
- ☐ keine

Aufgabe 2: Wo ist der BOB3 **berührungsempfindlich**?
Kreuze die richtige Antwort an:

- ☐ an den Armen
- ☐ am Kopf
- ☐ am Bauch
- ☐ an den Füßen
- ☐ nirgendwo
- ☐ an den LEDs

Aufgabe 3: Womit kann der BOB3 einem anderen Bob eine Botschaft senden? Kreuze die richtige Antwort an:

- ☐ mit dem Kopf
- ☐ mit den RGB-LEDs
- ☐ mit den weißen LEDs
- ☐ mit dem IR-Empfänger
- ☐ mit der IR-Sende-LED
- ☐ mit den Füßen

Aufgabe 4: Wie viele **Tastsensoren** hat der BOB3 insgesamt? Also an wie vielen **unterschiedlichen Stellen** kann der Bob Berührungen bemerken?
Kreuze die richtige Antwort an:

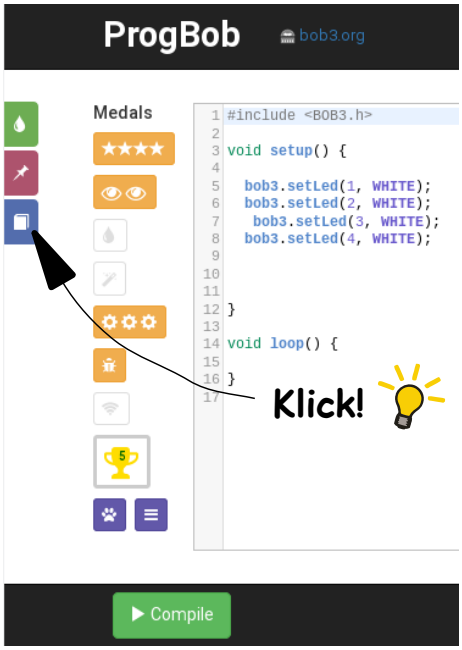
- ☐ ein
- ☐ zwei
- ☐ vier
- ☐ sechs
- ☐ acht
- ☐ zehn

Aufgabe 5: Wie kann der BOB3 **nah** und **fern** unterscheiden?
Kreuze die richtige Antwort an:

- ☐ er misst die Temperatur
- ☐ er leuchtet mit den hellen weißen LEDs und misst, wie weit er gucken kann
- ☐ mit dem IR-Sensor, per Reflektionsverfahren
- ☐ er weiß es nicht, er würfelt
- ☐ er fragt den Mikrocontroller
- ☐ er fragt einen anderen BOB3

Arbeitsblatt 7 - Bob's Software

Aus welchen Bestandteilen besteht die Bibliothek von BOB3?



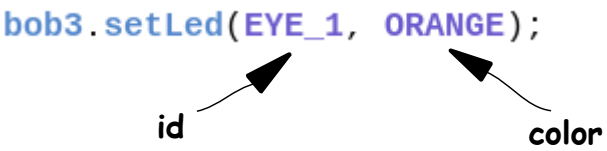
Wenn du im Programmier-Tutorial links neben den Medaillen auf den blauen Slide „**Bibliothek**“ klickst, dann bekommst du eine **Übersicht** über die **verschiedenen Methoden**, die du auf den BOB3 anwenden kannst.

Hier sind alle **Funktionen** zu finden, die schon **fertig implementiert** sind und einfach verwendet werden können, wie z.B. die Funktion `delay(500)`, die du einfach aufrufen und damit eine Verzögerung von 500 Millisekunden erzeugen kannst.

Funktionen, die sich auf ein **Objekt** beziehen, nennt man **Methoden**. Da der BOB3 im programmiertechnischen Sinne ein Objekt ist, heißen seine Funktionen Methoden.

Wenn du mit dem Mauszeiger **auf einer Methode wartest**, zeigt ein Info-Fenster alle Details zu der Methode an:

Das Beispiel zeigt die Details zu der Methode `bob3.setLed(id, color)`. Die Methode hast du schon oft verwendet, z.B. im Intro-I Tutorial:



Das Info-Fenster beschreibt zunächst, was die Methode macht: In unserem Beispiel wird die Farbe für eine LED gesetzt, es wird also eine LED in einer bestimmten Farbe eingeschaltet. Unsere Methode hat zwei **Parameter**: `id` und `color`.

Im Info-Fenster ist aufgelistet, was du alles für den **Parameter** `id` einsetzen darfst:

- Die Nummer der einzuschaltenden LED, also **1**, **2**, **3**, **4** oder auch
- Den Namen der einzuschaltenden LED, also **EYE_1**, **EYE_2**, **LED_3**, **LED_4**

Im Info-Fenster steht auch, was du alles für den **Parameter** `color` einsetzen darfst:

- Die Farbkonstante der einzuschaltenden LED, also z.B. **RED**, **BLUE**, **ORANGE**, **YELLOW** …etc.

Aufgabe 1: Wie viele **Parameter** hat die Funktion `delay(milliseconds)`?
Kreuze die richtige Antwort an:

- ☐ keinen
- ☐ einen
- ☐ zwei
- ☐ eintausend

Aufgabe 2: Wie viele **Parameter** hat die Methode `bob3.getTemperature()`?
Kreuze die richtige Antwort an:

- ☐ keinen
- ☐ einen
- ☐ zwei
- ☐ eintausend

Aufgabe 3: Kann man `bob3.setLed(Auge1, ORANGE)` schreiben, um das Auge 1 vom BOB3 orange einzuschalten?
Kreuze die richtige Antwort an:

- ☐ Ja, das klappt!
- ☐ Nein, das funktioniert nicht!
- ☐ Nein, das klappt nur manchmal...

Aufgabe 4: Kann man `bob3.setLed(1, ORANGE)` schreiben, um das Auge 1 vom BOB3 orange einzuschalten?
Kreuze die richtige Antwort an:

- ☐ Ja, das klappt!
- ☐ Nein, das funktioniert nicht!
- ☐ Nein, das klappt nur manchmal...

Aufgabe 5: Was darf man bei der Methode `bob3.setWhiteLeds(status1, status2)` für den Parameter `status1` einsetzen?
Kreuze die richtigen Antworten an, es sind mehrere Antworten richtig:

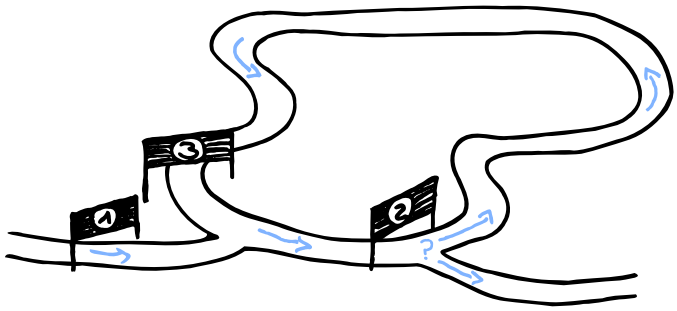
- ☐ true
- ☐ 0
- ☐ ON
- ☐ false
- ☐ WHITE
- ☐ OFF
- ☐ 1

Aufgabe 6: Was liefert die Methode `bob3.getTemperature()`, wenn man sie aufruft?
Kreuze die richtige Antwort an:

- ☐ entweder eine 0 oder eine 1
- ☐ die aktuelle Temperatur als Zahlenwert zwischen 0 und 255
- ☐ die aktuelle Temperatur als Wort, z.B. kalt, kälter, warm...
- ☐ da die Methode keinen Parameter hat, liefert sie nichts!

Arbeitsblatt 8 – for-Schleife

Eine for-Schleife dient zur wiederholten Durchführung:



Die Kontrollstruktur **for-Schleife** ist eine **Zähl-Schleife**. Sie ermöglicht, dass in Abhängigkeit von einer **Bedingung** bestimmte Anweisungen **solange immer wieder** ausgeführt werden, bis die Bedingung **nicht mehr** erfüllt ist.

```
for (Initialisierung; Bedingung; Aktualisierung) {
  Anweisungen;
}
```

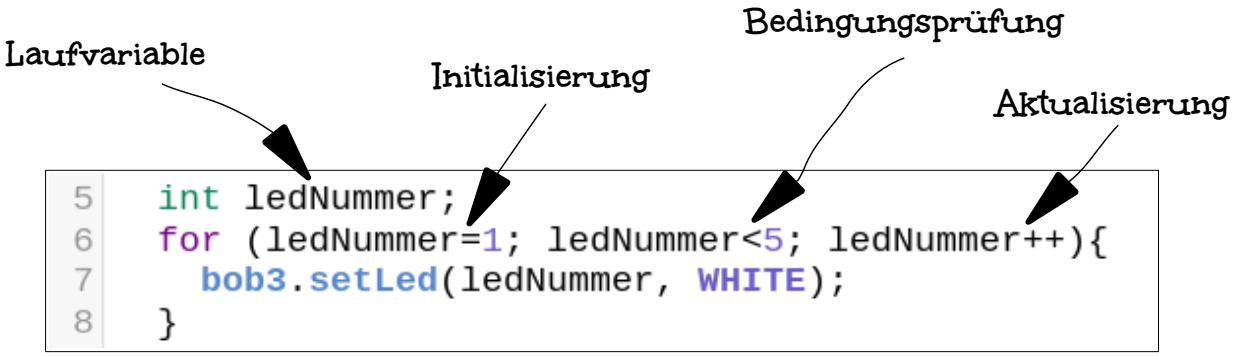
Die Parameterliste einer for-Schleife besteht aus **drei Teilen**:

Initialisierung → Die **Laufvariable** wird auf einen **Anfangswert** gesetzt
(Dies erfolgt nur bei der ersten Ausführung der Schleife!)

Bedingungsprüfung → Die Bedingung legt das **Abbruchkriterium** der Schleife fest

Aktualisierung → Am Ende eines Durchlaufs wird die **Laufvariable aktualisiert**

Wir schauen uns mal ein konkretes Beispiel an:



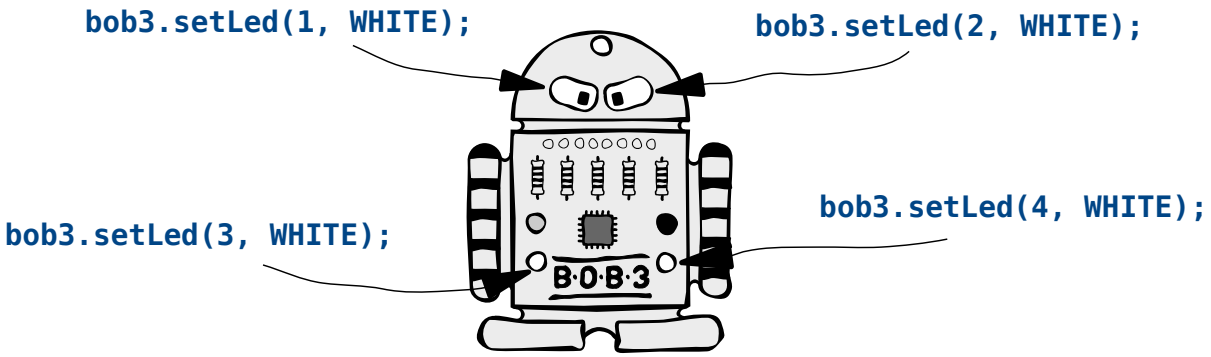
In **Zeile 5** wird unsere **Laufvariable** `ledNummer` deklariert, also neu eingeführt. Sie ist vom Typ `int`, kann also ganze Zahlen speichern.

In **Zeile 6** beginnt unsere for-Schleife, wir starten unseren **ersten Durchlauf**:

Schritt 1 – Initialisierung:	Die Variable `ledNummer` wird mit 1 initialisiert, also auf den Wert 1 gesetzt.
Schritt 2 – Bedingungsprüfung:	Wir überprüfen unsere Bedingung: Ist der Wert von `ledNummer` kleiner als 5?
Schritt 3 – Ausführung:	**Falls** die Bedingung **wahr** ist, werden nun die Anweisungen ausgeführt. In unserem Beispiel ist dies die Anweisung `bob3.setLed(ledNummer, WHITE);`
Schritt 4 – Aktualisierung:	**Aktualisierung** der Variablen `ledNummer`. Der Ausdruck `ledNummer++` bedeutet, dass `ledNummer` um 1 erhöht wird.
Schritt 5 – Wiederholung:	**Wiederholung ab Schritt 2**. Wir starten also einen erneuten Durchlauf durch unsere Schleife, wobei der Wert der Variablen `ledNummer` jetzt 2 ist.

‼ Es werden dann **solange** die Schritte 2 bis 5 abgearbeitet, bis die Bedingungsprüfung **falsch** ergibt ‼

Was genau macht unser Beispiel-Programm? Überlege mal!



→ Das Programm schaltet nacheinander alle vier LEDs vom BOB3 weiß an!

1. Durchlauf: **ledNummer ist 1**
Bedingung (1<5) ist erfüllt → **bob3.setLed(1, WHITE);**
setze ledNummer auf 2

2. Durchlauf: **ledNummer ist 2**
Bedingung (2<5) ist erfüllt → **bob3.setLed(2, WHITE);**
setze ledNummer auf 3

3. Durchlauf: **ledNummer ist 3**
Bedingung (3<5) ist erfüllt → **bob3.setLed(3, WHITE);**
setze ledNummer auf 4

4. Durchlauf: **ledNummer ist 4**
Bedingung (4<5) ist erfüllt → **bob3.setLed(4, WHITE);**
setze ledNummer auf 5

5. Durchlauf: **ledNummer ist 5**
Bedingung (5<5) ist nicht erfüllt!!
5<5 ist falsch! → **ABBRUCH der Schleife**

Aufgabe 1: Betrachte das folgende Programm. Wie oft wird die Anweisung **bob3.setLed(ledNummer, WHITE);** ausgeführt?

```
5  int ledNummer;
6  for (ledNummer=1; ledNummer<5; ledNummer++){
7    bob3.setLed(ledNummer, WHITE);
8  }
```

- ☐ einmal
- ☐ zweimal
- ☐ dreimal
- ☐ viermal
- ☐ fünfmal

Aufgabe 2: Betrachte das folgende Programm. Wie oft wird die Anweisung **bob3.setLed(ledNummer, WHITE);** ausgeführt?

```
5  int ledNummer;
6  for (ledNummer=1; ledNummer<4; ledNummer++){
7    bob3.setLed(ledNummer, WHITE);
8  }
```

- ☐ einmal
- ☐ zweimal
- ☐ dreimal
- ☐ viermal
- ☐ fünfmal

Aufgabe 3: Betrachte das folgende Programm. Wie oft wird die Anweisung **bob3.setLed(ledNummer, WHITE);** ausgeführt?

```
5   int ledNummer;
6   for (ledNummer=3; ledNummer<4; ledNummer++){
7     bob3.setLed(ledNummer, WHITE);
8   }
```

- ☐ einmal
- ☐ zweimal
- ☐ dreimal
- ☐ viermal
- ☐ fünfmal

Aufgabe 4: Betrachte das folgende Programm. Wie oft wird die Anweisung **bob3.setLed(ledNummer, WHITE);** ausgeführt?

```
5   int ledNummer;
6   for (ledNummer=4; ledNummer>0; ledNummer--){
7     bob3.setLed(ledNummer, WHITE);
8   }
```

- ☐ einmal
- ☐ zweimal
- ☐ dreimal
- ☐ viermal
- ☐ fünfmal

Aufgabe 5: Betrachte die folgende for-Schleife. Welche Werte nimmt die Variable **i** im jeweiligen Durchlauf an?

```
6   for (i=0; i<10; i=i+2){
7     ...
8   }
```

Durchlauf 1: **i =** _____

Durchlauf 2: **i =** _____

Durchlauf 3: **i =** _____

Durchlauf 4: **i =** _____

Durchlauf 5: **i =** _____

Aufgabe 6: Betrachte die folgende for-Schleife. Welche Werte nimmt die Variable **j** im jeweiligen Durchlauf an?

```
6   for (j=1; j<64; j=j*2){
7     ...
8   }
```

Durchlauf 1: **j =** _____

Durchlauf 2: **j =** _____

Durchlauf 3: **j =** _____

Durchlauf 4: **j =** _____

Durchlauf 5: **j =** _____

Durchlauf 6: **j =** _____