Arduino – eigene Befehle programmieren

Vermutlich sieht dein Programm zur Ampelsteuerung mit dem Blinken des grünen Lichts (vgl. Aufgabe 3) von Aufgabenblatt **ST_AA_05**) so oder so ähnlich aus:

```
int redLedPin = 10;
int yellowLedPin = 5;
int greenLedPin = 3;

void setup() {
  pinMode(redLedPin,OUTPUT);
  pinMode(yellowLedPin,OUTPUT);
  pinMode(greenLedPin,OUTPUT);
  //set inital LED-states
  digitalWrite(redLedPin,HIGH);
  digitalWrite(yellowLedPin,LOW);
  digitalWrite(greenLedPin,LOW);
}

void loop() {
  delay(5000);
  digitalWrite(redLedPin,LOW);
  digitalWrite(yellowLedPin,HIGH);
  delay(2000);
  digitalWrite(greenLedPin,HIGH);
  digitalWrite(yellowLedPin,LOW);
  delay(5000);
  digitalWrite(greenLedPin,LOW);
  delay(1000);
  digitalWrite(greenLedPin,HIGH);
  delay(1000);
  digitalWrite(greenLedPin,LOW);
  delay(1000);
  digitalWrite(greenLedPin,HIGH);
  delay(1000);
  digitalWrite(greenLedPin,LOW);
  delay(1000);
  digitalWrite(greenLedPin,HIGH);
  delay(1000);
  digitalWrite(greenLedPin,LOW);
  digitalWrite(yellowLedPin,HIGH);
  delay(2000);
  digitalWrite(yellowLedPin,LOW);
  digitalWrite(redLedPin,HIGH);
}
```

Dabei macht der Code für das dreimalige Blinken des grünen Lichts, also der grünen Leuchtdiode, einen recht großen Teil des `loop`-**Programmblock**s aus: Zuerst wird die grüne LED ausgeschaltet und dann dreimal ein- und wieder ausgeschaltet:

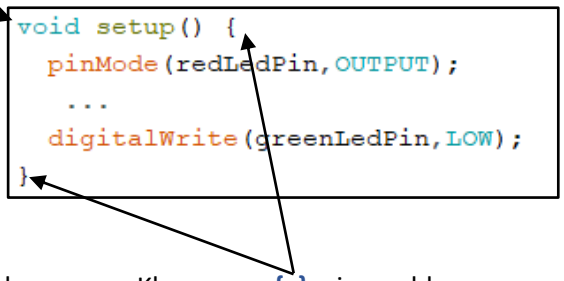
Wie praktisch wäre es doch, einen eigenen Befehl – z.B. greenLed3Blink() – dafür zu haben!

Um einen neuen Befehl selbst zu programmieren (und damit wortwörtlich einen „eigenen“ Befehl zu erschaffen) sind zunächst auch nur ein paar Dinge zu wissen bzw. zu beachten – welche das sind, wird auf der nächsten Seite erklärt:

Ein selbst programmierter, **eigener Befehl** ist ein **Programmblock**, ähnlich wie der `setup`-oder der `loop`-**Programmblock** und besteht wie diese

- aus einer **Befehlsüberschrift**, entsprechend (z.B.) `void setup()`.

  Diese Befehlsüberschrift selbst besteht (bis auf Weiteres) aus
  - dem Schlüsselwort `void`,
  - dem Namen des Befehls
  - einem Paar runder Klammern



- und dem **Befehlscode**, der als **Block** in ein paar geschwungene Klammern, {, }, eingeschlossen wird (vgl. Abbildung).

Der Rahmen für einen selbst programmierten Befehl sieht also zum Beispiel so aus:

```
void meinBefehl(){

}
```

Dieser Befehl wird dann – z.B. im `loop`-**Programmblock** – so verwendet:

```
void loop(){
  ...
  meinBefehl();
  ...
}
```

Bedeutsam ist dabei auch das Paar runder Klammern, das direkt nach dem Befehlsnamen folgt. Von diesen beiden Klammern werden die Daten eingeschlossen, die der Befehl „von außen“ benötigt. Wenn die runden Klammern – so wie in obigem Beispiel – leer sind, benötigt der Befehl keine solchen Daten, wenn jedoch der Rahmen für einen selbst programmierten Befehl z.B. so aussieht:

```
void meinAndererBefehl(int data){

}
```

dann darf im Code dieses Befehls `data` wie eine Variable verwendet werden. Der ganzzahlige Wert von `data` muss allerdings beim Aufruf dieses Befehls zwischen die runden Klammern geschrieben werden, z.B. so:

```
void loop(){
  int zahl = 7;
  ...
  meinAndererBefehl(7);
  //oder:
  meinAndererBefehl(zahl);
  ...
}
```