

Die Schleifenfunktion

Da wir die LED wiederholt blinken lassen wollen, müssen wir eine Schleifenfunktion (loop) erstellen, damit der Arduino eine Anweisung immer und immer wieder ausführt, bis der Strom abgeschaltet oder die Reset-Taste betätigt wird.

Geben Sie den fett gedruckten Code ein, der im folgenden Listing hinter dem Abschnitt von void setup() steht, um eine leere Schleifenfunktion zu erstellen. Achten Sie darauf, dass der neue Abschnitt mit einer schließenden geschweiften Klammer (}) endet, und speichern Sie den Sketch.

```
/*
  Arduino Blink LED Sketch
  by Mary Smith, created 09/09/12
  */

void setup()
{
  pinMode(13, OUTPUT); // Legt den digitalen Pin 13 als Ausgang fest
}

void loop()
{
  // Hierhin kommt der Code für die Schleife
}
```

WARNUNG

Die Arduino-IDE führt keine automatische Zwischenspeicherung Ihrer Skette durch, weshalb Sie regelmäßig speichern sollten!

Als Nächstes geben Sie in void loop() die Funktionen ein, die der Arduino ausführen soll.

Schreiben Sie den folgenden Code zwischen die geschweiften Klammern der loop-Funktion und klicken Sie dann auf *Verify*, um sicherzustellen, dass Sie dabei keinen Fehler gemacht haben.

```
digitalWrite(13, HIGH);    // Schaltet den digitalen Pin 13 ein
delay(1000);              // Wartet eine Sekunde lang
digitalWrite(13, LOW);    // Schaltet den digitalen Pin 13 aus
delay(1000);              // Wartet eine Sekunde lang
```

Das wollen wir uns im Einzelnen ansehen. Die Funktion `digitalWrite()` legt die Spannung fest, die an einem digitalen Pin ausgegeben wird, in diesem Fall Pin 13 für die LED. Der zweite Parameter dieser Funktion lautet `HIGH`, wodurch eine »hohe« Spannung ausgegeben wird. Der Strom fließt von dem Pin zur LED und schaltet diese ein. (Wenn Sie den Parameter auf `LOW` setzen, wird der Stromfluss zur LED unterbrochen.)

Nachdem die LED eingeschaltet ist, leuchtet sie eine Sekunde lang, was durch `delay(1000)` erreicht wird. Die Funktion `delay()` sorgt dafür, dass in dem angegebenen Zeitraum (in unserem Fall also 1000 Millisekunden oder eine Sekunde) nichts geschieht.

Danach wird die Spannung an der LED mit `digitalWrite(13, LOW)`; wieder ausgeschaltet. Abermals warten wir mit `delay(1000)` eine Sekunde lang. In diesem Zeitraum bleibt die LED ausgeschaltet.

Der vollständige Sketch sieht wie folgt aus:

```
/*
Arduino Blink LED Sketch
by Mary Smith, created 09/09/12
*/

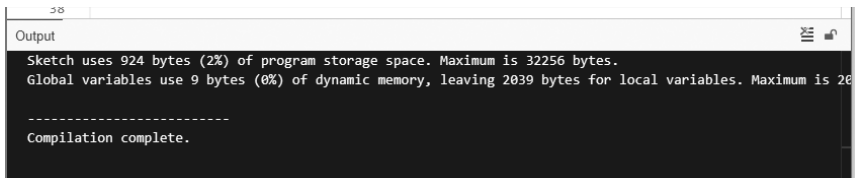
void setup()
{
  pinMode(13, OUTPUT);    // Legt den digitalen Pin 13 als Ausgang fest
}

void loop()
{
  digitalWrite(13, HIGH); // Schaltet den digitalen Pin 13 ein
  delay(1000);            // Wartet eine Sekunde lang
  digitalWrite(13, LOW);  // Schaltet den digitalen Pin 13 aus
  delay(1000);            // Wartet eine Sekunde lang
}
```

Bevor Sie weiterarbeiten, speichern Sie den Sketch!

Den Sketch überprüfen

Bei der Überprüfung wird untersucht, ob der Sketch auf eine Weise geschrieben ist, die der Arduino verstehen kann. Dazu klicken Sie in der IDE auf *Verify* und warten einen Augenblick. Nach der Überprüfung erscheint im Meldungsbereich eine Mitteilung wie in Abbildung 2–13.



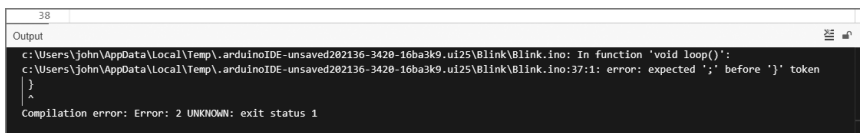
```
Output
Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 bytes for local variables. Maximum is 2048 bytes.

-----
Compilation complete.
```

Abb. 2–13 Der Sketch wurde überprüft.

Die Meldung »Done compiling« (Kompilierung erfolgt) besagt, dass der Sketch in Ordnung ist und auf den Arduino hochgeladen werden kann. Außerdem wird angezeigt, wie viel Arbeitsspeicher er belegt (hier 924 Byte) und wie viel auf dem Arduino insgesamt zur Verfügung steht (32.256 Byte).

Was aber, wenn der Sketch nicht korrekt ist? Nehmen wir beispielsweise an, Sie haben das Semikolon am Ende der zweiten `delay(1000)`-Funktion vergessen. Wenn Sie auf *Verify* klicken und irgendetwas in Ihrem Sketch nicht stimmt, wird eine Fehlermeldung wie die in Abbildung 2–14 ausgegeben.



```
38
Output
c:\Users\john\AppData\Local\Temp\arduinoIDE-unsaved202136-3420-16ba3k9.ui25\Blink\Blink.ino: In function 'void loop()':
c:\Users\john\AppData\Local\Temp\arduinoIDE-unsaved202136-3420-16ba3k9.ui25\Blink\Blink.ino:37:1: error: expected ';' before '}' token
}
^
Compilation error: Error: 2 UNKNOWN: exit status 1
```

Abb. 2–14 Ein Überprüfungsfehler im Meldungsbereich

Die Meldung gibt an, dass der Fehler in der Funktion *void loop* steckt. Außerdem wird angezeigt, dass der Fehler selbst ein fehlendes Semikolon ist mit der Ausgabe `error: expected ';' before '}' token`. Die IDE markiert zudem die Position des Fehlers oder eine Stelle gleich dahinter. Dadurch können Sie den Fehler schnell finden und beheben.

Den Sketch hochladen und ausführen

Wenn Sie sich vergewissert haben, dass der Sketch korrekt ist, speichern Sie ihn. Stellen Sie sicher, dass der Arduino angeschlossen ist, und klicken Sie in der IDE auf *Upload*. Möglicherweise überprüft die IDE den Sketch erneut, lädt ihn dann aber auf den Arduino hoch. Während dieses Vorgangs blinken die LEDs TX und RX auf der Platine (siehe Abb. 2–6), um anzuzeigen, dass Informationen zwischen dem Arduino und dem Computer übertragen werden.

Jetzt schlägt die Stunde der Wahrheit: Der Arduino sollte damit beginnen, den Sketch auszuführen. Wenn alles richtig funktioniert, sollte die LED im Sekundentakt blinken.

Herzlichen Glückwunsch! Sie kennen jetzt die Grundlagen, um Arduino-Sketche erstellen, überprüfen und hochladen zu können.

Den Sketch bearbeiten

Nachdem Sie den Sketch ausgeführt haben, können Sie ihn abwandeln, indem Sie beispielsweise die Verzögerung zum Ein- und Ausschalten der LED ändern. Da die IDE ähnlich funktioniert wie eine Textverarbeitung, können Sie den gespeicherten Sketch einfach öffnen, die Werte anpassen, den Sketch wieder speichern und erneut auf den Arduino hochladen. Um beispielsweise den Blinkrhythmus zu beschleunigen, ändern Sie die beiden `delay`-Funktionen auf einen Wert von 250, sodass die LED im Viertelsekundentakt aufleuchtet und erlischt.

```
delay(250); // Wartet eine Viertelsekunde lang
```

Laden Sie den Sketch erneut hoch. Die LED sollte jetzt schneller blinken und jeweils eine Viertelsekunde lang ein- und ausgeschaltet sein.

Ausblick

Mit Ihren neuen Kenntnissen darüber, wie Sie Arduino-Sketche schreiben, bearbeiten, speichern und hochladen, sind Sie nun bereit, in den nächsten Kapiteln zu lernen, wie Sie weitere Funktionen einsetzen, Ihre Projekte sauber gestalten, einfache elektronische Schaltkreise konstruieren usw.