Es lässt sich jetzt allerdings sehr einfach eine weitere Blinkschaltung realisieren, indem man die LOW-delay-Zeiten beider LEDs auf 0 setzt oder die LOW-delay-Zeiten ganz löscht.

- Setzen Sie beide LOW-delay-Zeiten auf 0 ms, kompilieren und laden Sie neu.
- Speichern Sie diesen Sketch unter *Blinkschaltung_3*.

Funktionsweise

Die LED 1 ist 0,5 s an und schaltet dann aus, gleichzeitig wird die LED 2 für 2 s eingeschaltet. Dann schaltet die LED 2 aus und die LED 1 wieder für 0,5 s ein usw.

Deklaration und Initialisierung von Variablen

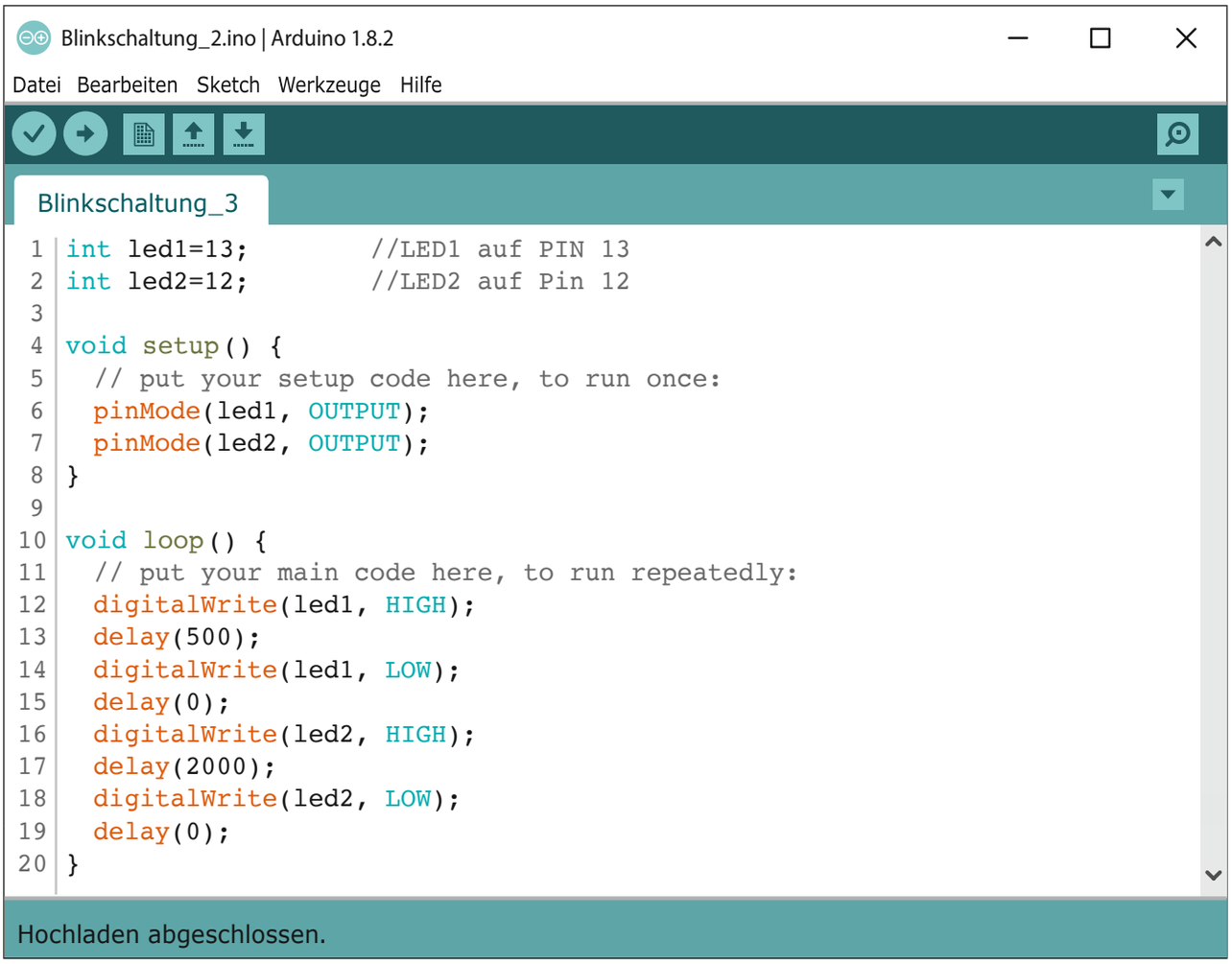
Durch Verwendung von Variablen (Info 5) können z. B. Zuweisungen leicht geändert werden. Die Variablen können innerhalb ihres Wertebereiches unterschiedliche Werte annehmen. Die Variable *int* (Integer) kann ganze Zahlen zwischen –32.768 bis +32.767 annehmen.

Dazu müssen diese Zuordnungen als solche deklariert werden und stehen vor dem *void setup*. Die Variable besteht z. B. aus:

dem Typ: *int*, dem Namen: *led1* und einem Wert: *13*.

z. B. Zuordnung: *int led1=13* d. h. Led 1 liegt auf Pin 13

Programm mit Variablen



Im Gegensatz zum vorherigen Programm stehen jetzt in den Befehlen im *void setup ()* und in der *void loop ()* nicht mehr die Pin-Nummern 12 bzw. 13, sondern *led1* bzw. *led2*.

Also statt *pinMode(13, OUTPUT);* jetzt *pinMode(led1, OUTPUT);*

Zum einen wird das Programm dadurch leichter lesbar, zum anderen kann man die Zuordnungen leicht ändern.

Speichern Sie diesen Sketch unter *Blinkschaltung_4* ab.

Sollen z. B. statt der Pins 12 und 13 die Pins 6 und 7 verwendet werden, so muss man dies nur in der Initialisierung ändern. Diese Änderung gilt dann für das ganze Programm. Man nennt solche Variablen **globale Variablen**.

Es können auch Variablen innerhalb einer Schleife oder einer Funktion deklariert werden. Solche **lokalen Variablen** gelten dann nur innerhalb des eigenen Codeblocks, also innerhalb der geschweiften Klammern { }, in der die Deklaration steht.

Bei **konstanten Variablen** wird vor die Variable ein *const* geschrieben, also z. B. statt *int* steht jetzt *const int*. Durch das *const* wird verhindert, dass die Variable im Laufe des Programms einen neuen Wert erhalten kann. Nach der Deklaration und Initialisierung kann die Variable nur noch ausgelesen werden. An der Funktion des Programms ändert sich nichts.

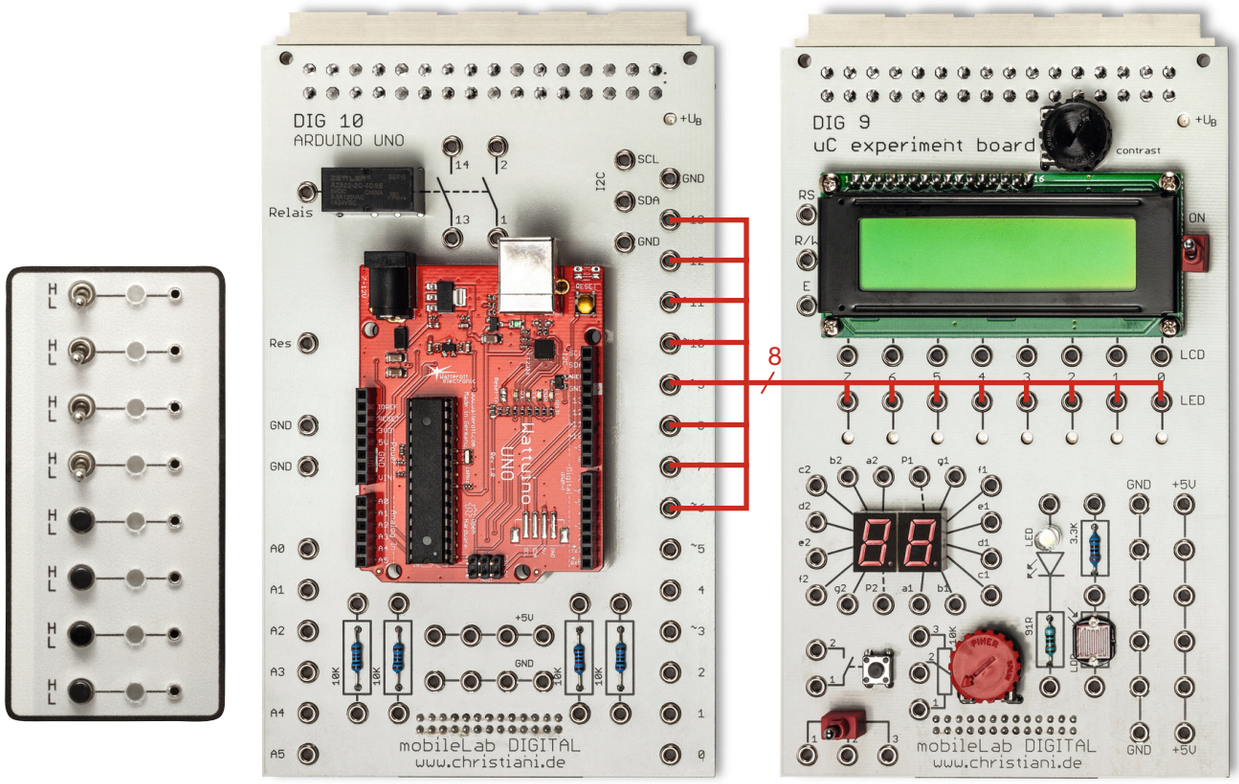
Übung

Ersetzen Sie im vorherigen Programm die Variable *int* durch *const int* und überprüfen Sie die Funktion.

2.1.4 Einfaches Lauflicht mit 8 LEDs

Versuchsaufbau

Erweitern Sie die Schaltung um 6 zusätzliche LEDs, ergänzen Sie den Verdrahtungsplan und legen Sie die *LEDs 0 – 7* auf die *Pins 13 – 6*.



Aufgaben

- Ergänzen Sie das Programm um die zusätzlichen LEDs und setzen Sie die Einschaltzeiten auf z. B. 200 ms.
- Kompilieren und laden Sie das Programm und speichern Sie diesen Sketch unter *Einfaches_Lauflicht* ab.

Lösung s. Anhang auf Seite 116

(Anmerkung: Eine elegantere Lösung finden Sie später unter „2.5.1.2 Segmenttest für eine 7-Segmentanzeige mit Array und Schleife“ auf Seite 61.)

2.1.5 Baustellenampel

An einer Baustelle soll der Verkehr in beiden Richtungen mit einer Baustellenampel über eine Fahrspur geleitet werden. Für dieses Beispiel werden die Durchlasszeiten (grün) und Sicherheitswartezeiten (beide Ampeln rot) verkürzt.

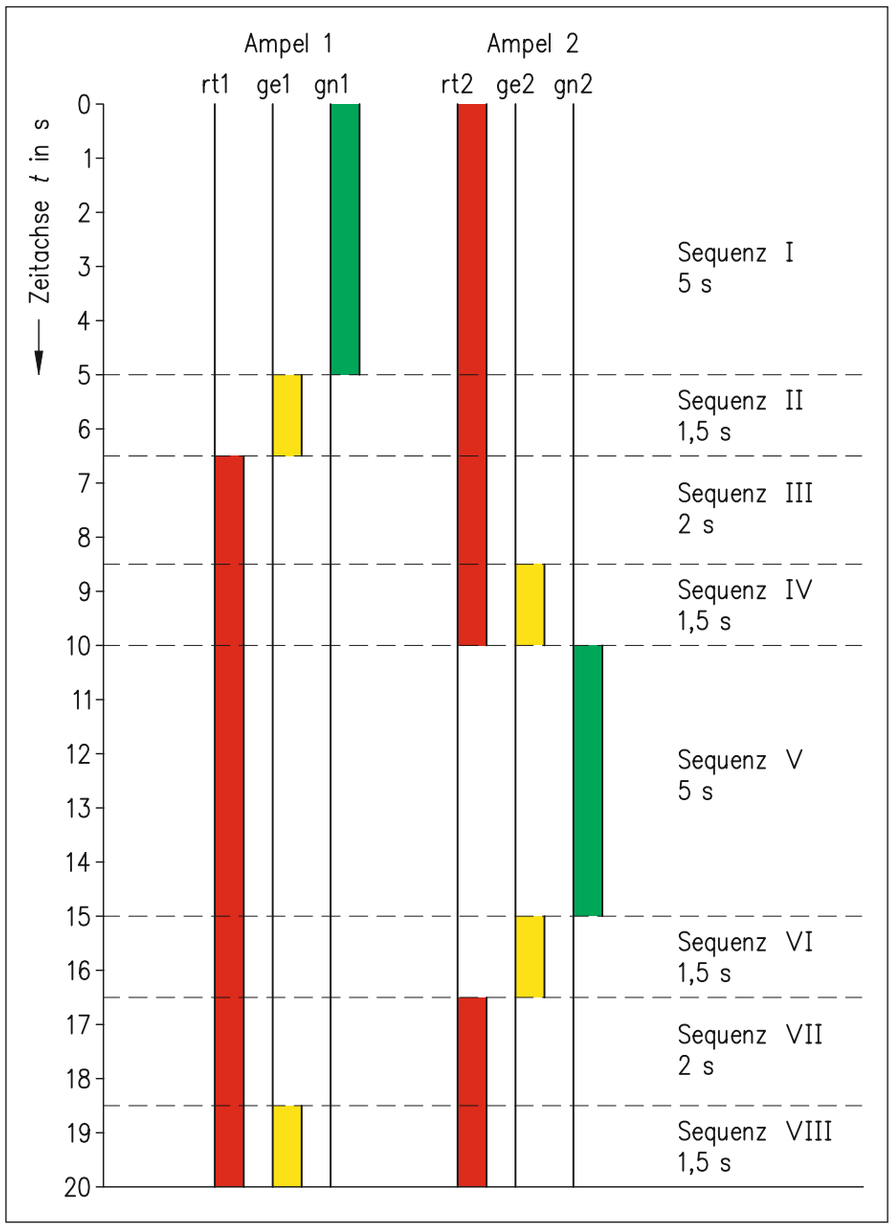
Die Durchlasszeit soll 5 s betragen und die Sicherheitswartezeit (beide Ampeln rot) 2 s. Die Gelbphase beträgt 1,5 s.

Aufgabe 1

- Ergänzen Sie den nachfolgenden Zeitablaufplan.
- Teilen Sie den Zeitablaufplan so in einzelne Sequenzen auf, dass bei jeder Änderung eine neue Sequenz beginnt.

Zeitablaufplan

rt = rot ge = gelb gn = grün



Die LEDs sollen folgendermaßen zugeordnet werden.

rt1(7) = 13; ge1(6) = 12; gn1(5) = 11;

rt2(0) = 10; ge2(1) = 9; gn2(2) = 8;

In Klammern () stehen die LED-Nummern.