Befehle neu

digitalWrite(13,value);	= Schreibe (setze) den Ausgang digital auf den Wert (value)
value=digitalRead(12);	= Der digitale Wert wird gelesen (gemessen)
value=!digitalRead(12);	= Der gelesene (gemessene) digitale Wert wird durch das Ausrufezeichen invertiert

Externe Pullup- und Pulldown-Widerstände werden insbesondere dann verwendet, wenn die internen Widerstände zu hoch sind. Das kann z. B. sein, wenn lange Anschlussleitungen verwendet werden, die dann wie Antennen wirken. Hier kann man den externen Pullup- oder Pulldown-Widerstand bis auf 5 kΩ verringern. Nachteilig ist, dass im geschalteten Zustand ein Strom über den Widerstand fließt und damit eine Verlustleistung erzeugt.

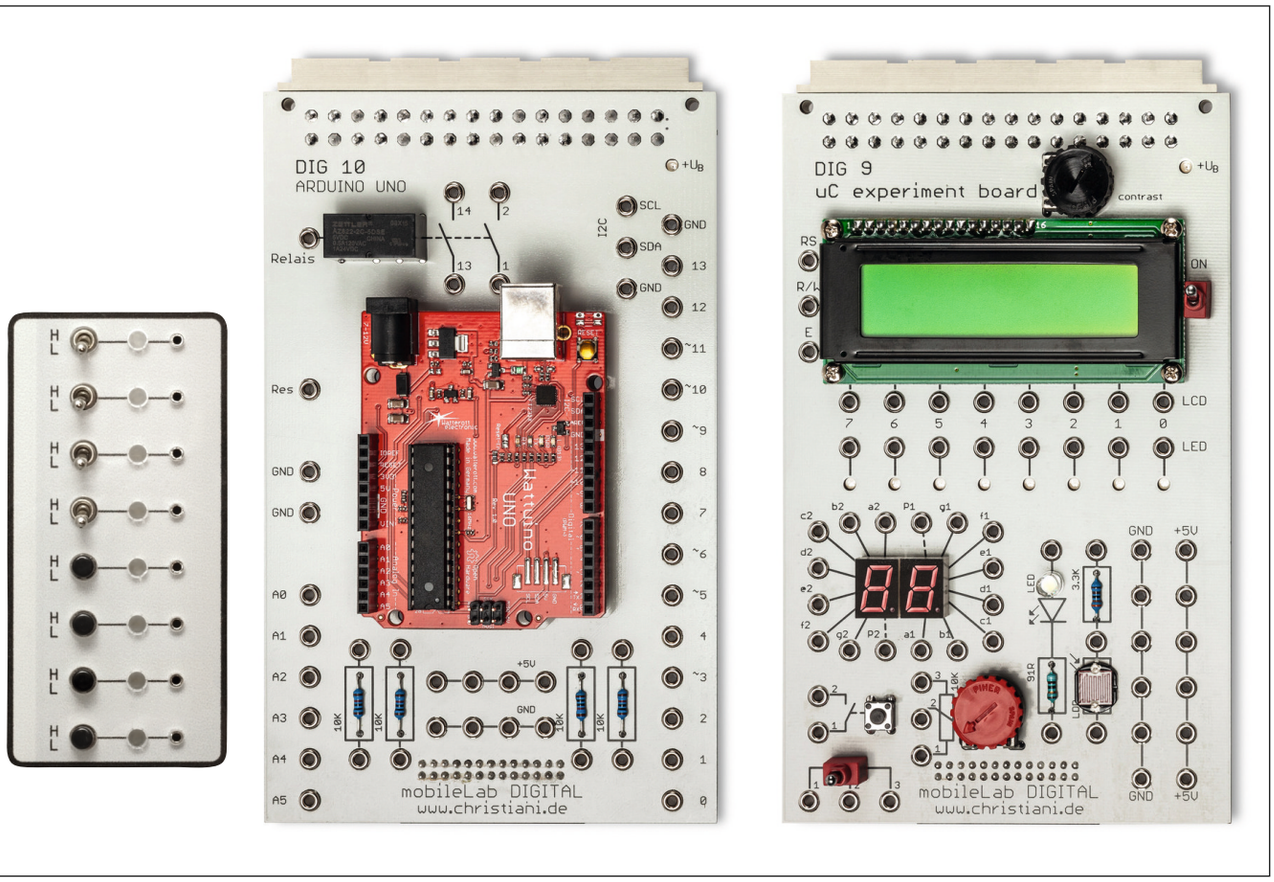
Beispiel $R = 5\ \text{k}\Omega$; $U = 5\ \text{V}$ Ges.: I, Pv

2.1.8 Aufgabe: Wechselschalter und zwei LEDs

Aufgabe

Mit dem Wechselschalter (10) auf dem *experiment board* sollen wechselweise zwei LEDs geschaltet werden. Der mittlere Anschluss 2 des Wechselschalters liegt auf GND, die Anschlüsse 1 und 3 werden mit den Pins 10 und 11 des Arduino verbunden. Die LEDs liegen an den Pins 12 und 13.

Versuchsaufbau

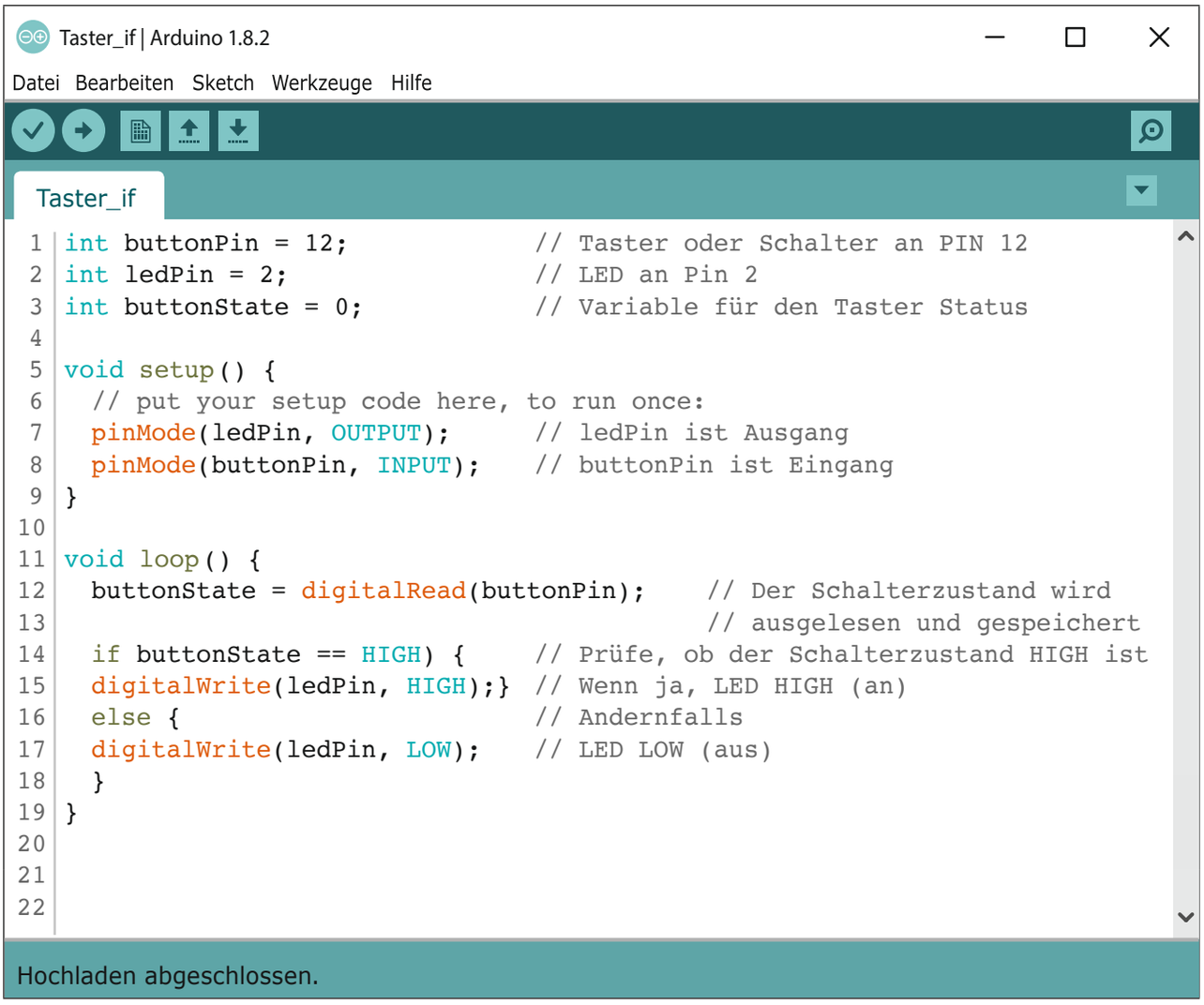


2.1.9 Auslesen von Taster oder Schalter mit *if*

Der Zustand eines Tasters oder Schalters soll mit der *if*-Funktion ausgelesen werden.

Aufgaben

- Schließen Sie an Pin 2 eine LED an.
- Verbinden Sie einen Schalter oder einen Taster von der Grundplatte mit Pin 12.
- Laden Sie den Sketch und überprüfen Sie die Funktion.



Erläuterung

Mit der *if*-Funktion wird abgefragt, ob der Zustand des Schalters HIGH ist:

Dann wird der Ausgang HIGH gesetzt (LED an). Anderenfalls *(else)* wird der Ausgang auf LOW gesetzt (LED aus).

Das doppelte Gleichheitszeichen == ist ein Vergleichsoperator (siehe Anhang: Tabelle 2).

Befehl neu

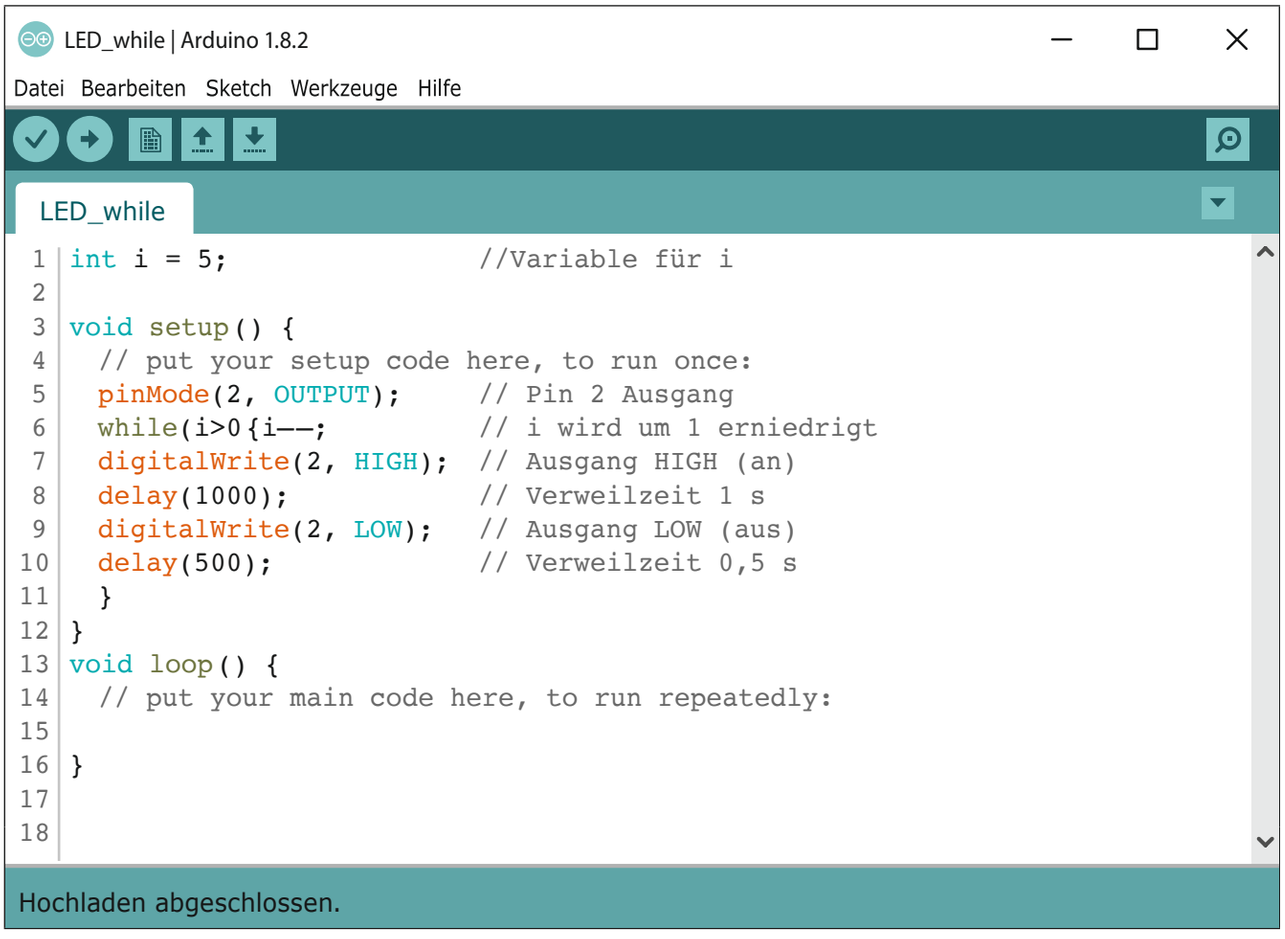
If-else	= wenn – anderenfalls	bzw.	entweder – oder

2.1.10 Wiederholungen mit *while*

Ein Vorgang, z. B. Blinken einer LED, soll 5-mal wiederholt werden.

Aufgaben

- Verbinden Sie eine LED mit Pin 2.
- Laden Sie den Sketch und überprüfen Sie die Funktion.



Erläuterung

Die Variable *i* ist maximal 5.

Mit *while(i>0)* { wird eine Schleife eingeleitet, die so lange läuft, wie die Bedingung $i > 0$ wahr ist. Im Codeblock der Schleife (der Code zwischen { und }) wird *i* mit *i*--; von 5 immer um 1 reduziert (Dekrement) }.

Aufgabe

Ändern Sie das Programm so ab, dass von $i = 0$ aufwärts gezählt wird.

2.1.11 LED dimmen, Analogeingang, serieller Monitor

LED dimmen, Analogeingang

Um LEDs zu dimmen, verwendet man in der Regel die Pulsweitenmodulation (PWM). Dabei wird nicht die Spannung mehr oder weniger reduziert, sondern man schaltet die LED mit einer festen Frequenz $f = \frac{1}{T}$ ein und aus und verändert die Einschaltdauer t_E. Ist die Einschaltzeit t_E z. B. genauso lang wie die Ausschaltzeit t_A, dann beträgt die Leistung *P* an der LED 50 %.

Die Flimmergrenze des Auges liegt bei etwa 18 Hz. Die Frequenz des PWM-Signals ist mit ca. 490 Hz so hoch, dass das Auge das Ein- und Ausschalten nicht sieht, sondern nur einen Mittelwert wahrnimmt.

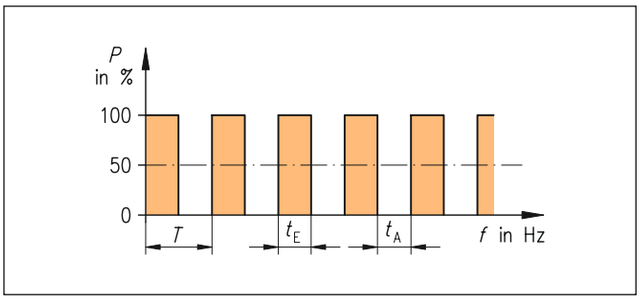


Bild 1 *Leistung P = 50 %*

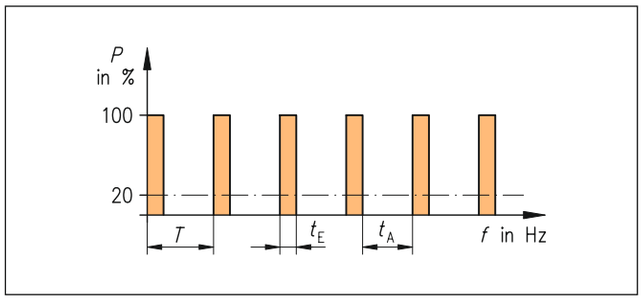


Bild 2 *Leistung P = 20 %*

PWM-Ausgänge

Nicht alle Pins liefern ein PWM-Ausgangssignal. Die PWM-Ausgänge Pin 3, 5, 6, 9, 10 und 11 sind durch ~ gekennzeichnet. Die Ausgänge beim Arduino können standardmäßig 8 Bit, d. h. 256 Werte (0 … 255) darstellen.

Analogeingänge

Analoge Signale kann man nur an die 6 Analogeingänge A0 – A5 anlegen. Diese Eingänge können auch als Digitaleingänge verwendet werden. Dann liegen am Eingang entweder 0 (0 V) oder 1 (5 V).

Bei einer Nutzung als Analogeingang ist eine Initialisierung nicht notwendig, der Wert kann direkt mit dem Befehl *analogRead* ausgelesen werden. Dennoch ist es sinnvoll, grundsätzlich jeden Pin, den man verwenden will, mit *pinMode( );* zu initialisieren.