# Leseprobe


Dr.-Ing. Paul Christiani GmbH & Co. KG
www.christiani.de

02

## BAUTEILE

TASTER

LED

220 OHM WIDERSTAND

10 KOHM WIDERSTAND

# RAUMSCHIFF-STEUERKONSOLE

## DEIN GENUINO WIRD ZUM STAR IN EINEM WELTRAUM-FILM

*Entdecke:* Digitale Ein- und Ausgabe, dein erstes Programm, Variablen
*Zeit:* **45 MINUTEN**
*Schwierigkeit:* ■□□□□
*Basiert auf den Projekten:* 1

*Nachdem du die Grundlagen der Elektrizität beherrschst, wird es Zeit, Dinge mit deinem Genuino zu steuern. In diesem Projekt wirst du eine Steuerkonsole bauen, die aussieht wie in einem Raumschiff aus einem Science-Fiction-Film der 70er Jahre. Deine Steuerkonsole wird aus einem Taster und Lichtern bestehen. Wenn du den Taster drückst, gehen die Lichter an. Du entscheidest, ob sie bedeuten: „Hyperdrive aktivieren" oder „Laser abfeuern!". Eine grüne LED leuchtet, bis du den Taster drückst. Sobald der Genuino ein Signal vom Taster bekommt, geht die grüne LED aus und zwei andere Lichter beginnen zu blinken.*

Die digitalen Pins des Genuino können nur mit zwei verschiedenen Zuständen umgehen: Entweder es liegt eine Spannung an oder eben nicht. Diese Art von Eingabe wird „digital" genannt (manchmal auch „binär"). Die beiden Zustände werden als HIGH (hoch) und LOW (niedrig) bezeichnet. HIGH bedeutet: „Eine Spannung liegt an". LOW heißt: „Es liegt keine Spannung an". Wenn du mit der Funktion `digitalWrite()` einen Ausgabe-Pin auf HIGH setzt, schaltest du ihn an. Prüfst du die Spannung zwischen dem Pin und der Masse, erhältst du 5 Volt. Setzt du den Ausgabe-Pin hingegen auf LOW, schaltest du ihn ab.
Die digitalen Pins des Genuinos können zur Eingabe (INPUT) und zur Ausgabe (OUTPUT) benutzt werden. Im Code konfigurierst du die Pins abhängig vom Zweck. Werden die Pins als Ausgabe benutzt, kannst du Bauteile wie LEDs einschalten. Werden die Pins für die Eingabe konfiguriert, kannst du damit prüfen, ob ein Taster gedrückt wird oder nicht. Die ersten beiden Pins 0 und 1 für den Datenaustausch werden mit dem Computer benutzt, beginne deshalb am besten mit Pin 2.

## DIE SCHALTUNG AUFBAUEN

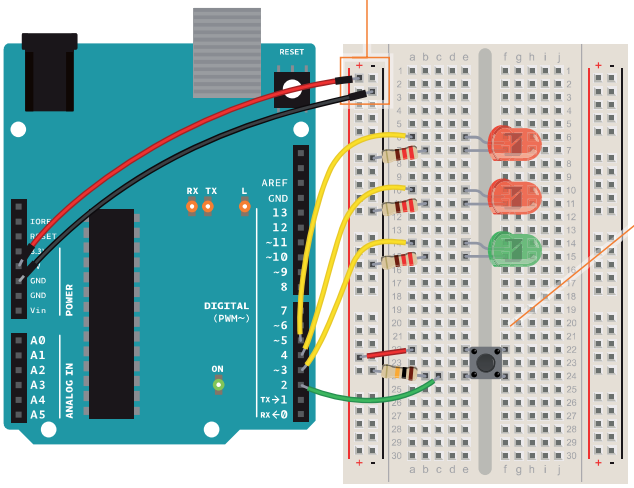



Abb. 1

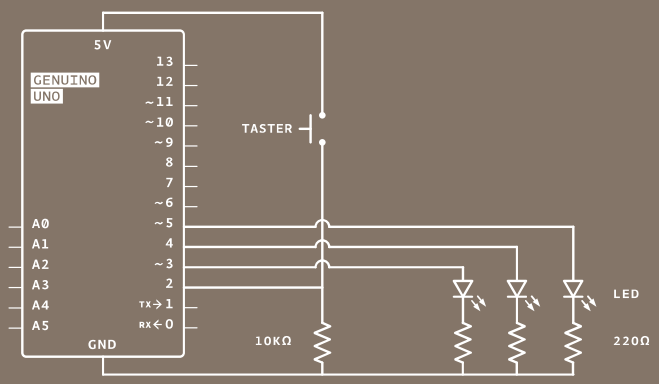
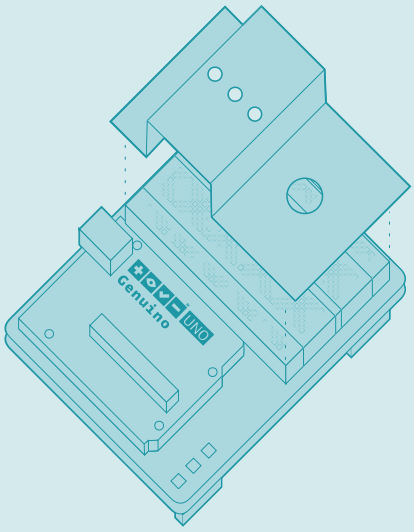



Abb. 2

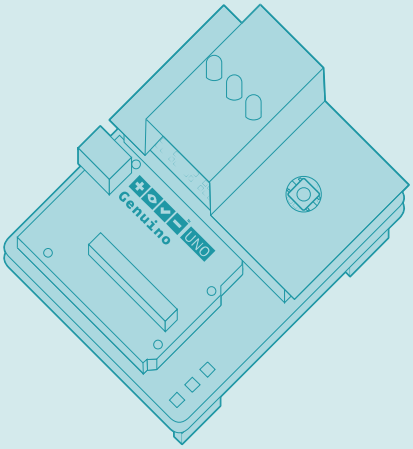
1 Verkabele dein Steckbrett mit dem 5-Volt- und dem Masse-Anschluss des Genuino wie im vorherigen Projekt. Stecke die beiden roten LEDs und eine grüne LED auf das Steckbrett. Verbinde die Kathode (kurzes Bein) jeder LED mit der Masse über einen 220-Ohm-Widerstand. Verbinde die Anode (langes Bein) der grünen LED mit Pin 3. Die Anoden der roten LED werden jeweils mit Pin 4 und 5 verbunden.

2 Stecke den Taster auf das Steckbrett wie im vorherigen Projekt. Verbinde eine Seite mit dem Stromanschluss und die andere Seite mit dem digitalen Pin 2 auf dem Genuino. Dann musst du das mit dem Genuino verbundene Bein des Tasters zusätzlich über einen 10-KOhm-Widerstand mit der Masse verbinden. Dieser sogenannte Pull-Down-Widerstand verbindet den Pin mit der Masse, wenn der Taster offen ist. Der Pin ist dann LOW, da der Taster keine Spannung leitet.

Du kannst das Steckbrett mit der Vorlage im Starterkit abdecken. Oder bastelst selber eine Dekoration für das Steuerpult. Wenn Lichter an und aus gehen, bedeutet das erst einmal nichts. Erst, wenn du sie in ein Gehäuse packst und mit Aufschriften versiehst, besagt ihr Verhalten etwas. Was soll die grüne LED bedeuten? Was die blinkenden roten LEDs? Du entscheidest!




1 Falte die ausgestanzte Pappe wie dargestellt.




2 Stülpe sie über das Steckbrett. Die drei LEDs und der Taster halten die Pappe an ihrem Ort.

## DER CODE

### Hinweise zum Start

Jedes Genuino-Programm hat zwei Hauptfunktionen. Funktionen sind Teile eines Programms, die bestimmte Kommandos ausführen. Funktionen haben eindeutige Namen und werden nur aufgerufen, wenn es notwendig ist. Ein Genuino-Programm erfordert mindestens die beiden Funktionen `setup()` und `loop()`. Diese Funktionen musst du deklarieren. Dann weiß der Genuino, was sie machen. Wie `setup()` und `loop()` deklariert werden, kannst du auf der rechten Seite sehen. In diesem Programm erstellst du eine Variable, bevor du zum Hauptteil des Programms kommst.
Variablen sind Orte im Speicher des Genuino, an denen du Werte hinterlegst. So merkt sich das Programm, was passiert ist. Der Wert einer Variablen hängt von den Anweisungen im Programm ab Variablennamen sollten den Wert beschreiben, den sie speichern. Eine Variable namens `switchState` (tasterZustand) besagt, dass sie den Zustand des Tasters speichert. Eine Variable namens „x" sagt hingegen kaum etwas über ihren Inhalt aus.

### Schreibe deinen ersten Code

Eine Variable wird zusammen mit der Typangabe erzeugt. Der Datentyp `int` speichert eine Ganzzahl (Englisch: Integer), das sind alle Zahlen ohne Komma. Weise ihr bei der Deklaration auch einen Wert zu. Die Deklaration wird mit einem Semikolon (;) abgeschlossen, wie jede andere Anweisung.

### Pin-Funktion konfigurieren

`setup()` wird einmal ausgeführt, sobald der Genuino mit Strom versorgt wird. Hier konfigurierst du mit der `pinMode()`-Funktion die digitalen Pins als Eingabe oder Ausgabe.
Die mit den LEDs verbundenen Pins werden zu Ausgabe-Pins (`OUTPUT`). Der Pin zum Taster wird zu einem Eingabe-Pin (`INPUT`).

### Schreibe die loop-Funktion

Die `loop()`-Funktion läuft regelmäßig, nachdem `setup()` durchlaufen wurde. Innerhalb von `loop()` prüfst du die Spannung an den Eingang-Pins und schaltest die Ausgang-Pins an oder aus. Um das Spannungsniveau an einem Eingang zu prüfen, benutzt du die Funktion `digitalRead()`. Sie prüft den angegebenen Pin auf eine anliegende Spannung. Der zu prüfende Pin wird als Argument übergeben. Argumente sagen einer Funktion, was oder womit sie etwas tun soll. Zum Beispiel benötigt `digitalRead()` das Argument, welcher Pin überprüft werden soll. In deinem Programm prüft `digitalRead()` den Zustand von Pin 2 und speichert den

```
void setup(){
}

void loop(){
}
```

**{ Geschweifte Klammern }**
Dein Code in den geschweiften Klammern wird ausgeführt, wenn die Funktion aufgerufen wird.

```
int switchState = 0;
```

```
void setup(){
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(2, INPUT);
}
```

**Groß- und Kleinschreibung**
Achte auf die korrekte Groß- und Kleinschreibung in deinem Code Zum Beispiel ist `pinMode` der Name einer Funktion, pinmode hingegen wird zu einem Fehler führen.

```
void loop(){
  switchState = digitalRead(2);
  // Das ist ein Kommentar
```

**Kommentare**
Wenn du etwas in normaler Sprache in deinem Programm schreiben willst, tust du das in einem Kommentar. Kommentare dienen als Erinnerung. Vom Computer werden sie ignoriert. Ein Kommentar beginnt mit zwei Schrägstrichen //
Der Computer wird alles ignorieren, was nach den Schrägstrichen steht.

Zustand in der Variable `switchState`. Wird `digitalRead()` aufgerufen, wenn eine Spannung an diesem Pin anliegt, erhält die Variable `switchState` den Wert `HIGH` (oder 1). Liegt keine Spannung am Pin an, erhält die Variable den Wert `LOW` (oder 0).

### Die If-Anweisung

Im obigen Code hast du das englische Wort if benutzt, um den Zustand von etwas (hier den Tasterzustand) zu prüfen. Eine `if()`-Anweisung vergleicht zwei Dinge miteinander und entscheidet, ob ein Vergleich wahr oder falsch ist. Je nachdem führt es dann deine Anweisungen aus. Wenn du in deinem Programm zwei Dinge miteinander vergleichst, benutze zwei Gleichheitszeichen ==. Mit einem einzelnen weist du hingegen einen Wert zu.

### Bau dein Raumschiff

`digitalWrite()` legt eine Spannung von 5 oder 0 Volt auf einen Ausgabepin an. Die Funktion erwartet zwei Argumente: den anzusteuernden Pin und die Spannung: `HIGH` oder `LOW`. Um in der `if()`-Anweisung die rote LED an und die grüne aus zu schalten, sollte der Code so aussehen.

*Wenn du das Programm ausführst, ändern sich die Lichter, wenn du den Taster drückst. Das ist ganz nett, aber du kannst es etwas interessanter gestalten, wenn du das Programm ausbaust.*

Du hast dem Genuino gesagt, was er tun soll, wenn der Taster offen ist. Definiere jetzt, was passiert, wenn der Taster gedrückt ist. Die `if()`-Anweisung hat eine optionale `else`-Anweisung. Sie wird ausgeführt, wenn die ursprüngliche Bedingung nicht erfüllt wurde. In diesem Fall prüfst du, ob der Taster nicht gedrückt (`LOW`) war. Deshalb schreibst du nach der `else`-Anweisung den Code für den Zustand `HIGH`.

Damit die rote LED blinkt, wenn der Taster gedrückt wurde, musst du im gerade geschriebenen `else`-Block die Lichter an- und ausschalten. Der Code sieht dafür wie hier aus.

*Jetzt wird dein Programm die roten LEDs blinken lassen, wenn der Taster gedrückt ist.*

Wurde der Zustand der LED geändert, sollte der Genuino einen Moment vor einer weiteren Änderung warten. Passiert das nicht, geht die LED sehr schnell an und aus. So wirkt es, als ob das Licht nur schwach leuchtet, denn `loop()` wird mehrere tausend Mal pro Sekunde aufgerufen. Die Schaltvorgänge erfolgen schneller, als du es wahrnimmst. Mit der `delay()`-Funktion hältst du eine Zeit lang den Genuino an, er führt dann keine weiteren Befehle aus. Du übergibst ihr als Argument die Wartedauer in Millisekunden. Eine Sekunde ist 1000 Millisekunden land. Der Aufruf von `delay(250)` entspricht einer Wartedauer von einer Viertelsekunde.

```
  if (switchState == LOW) {
  // Der Taster ist nicht gedrueckt
    digitalWrite(3, HIGH); // gruene LED
    digitalWrite(4, LOW);  // rote LED
    digitalWrite(5, LOW);  // rote LED
  }
  else {  // Der Taster ist gedrueckt
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
    delay(250); // Warte fuer eine Viertelsekunde
    // Schalte die LEDs um
    digitalWrite(4, HIGH);
    digitalWrite(5, LOW);
    delay(250); // Warte fuer eine Viertelsekunde
  }
} // Gehe zurueck zum Anfang der loop()-Funktion
```

Es kann hilfreich sein, den Programmablauf mit Pseudocode zu formulieren: Dabei beschreibst du das Programm mit normalen Worten. Deine Sätze sollten allerdings so aufgebaut sein, dass daraus das richtige Programm entstehen kann. Hier beschreibst du, was passiert, wenn der `switchState` `HIGH` ist (der Taster ist gedrückt) oder nicht. Wenn der Taster gedrückt ist, willst du die grüne LED ausschalten und die roten an. In Pseudocode sieht die Anweisung so aus:

```
wenn der switchState LOW
ist:
schalte die grüne LED an
schalte die roten LEDs aus

wenn der switchState HIGH
ist:
schalte die grüne LED aus
schalte die roten LEDs an
```

## ANWENDEN

Nachdem dein Genuino programmiert wurde, sollte das grüne Licht leuchten. Wenn du den Taster drückst, blinken die roten Lichter und das grüne Licht geht aus. Experimentiere mit den Zeitangaben in den beiden `delay()`-Funktionen. Verfolge was mit den Lichtern passiert und wie sich die Reaktion des Systems verändert - je nach Geschwindigkeit des Blinkens. Wenn du `delay()` in einem Programm aufrufst, pausieren alle anderen Funktionen. Es werden keine Sensordaten eingelesen, bis die Pause vorüber ist. Häufig sind solche Pausen nützlich. Doch überlege bei deinen Projekten immer, ob sie die Bedienung eines Systems nicht unnötigerweise beeinflussen.

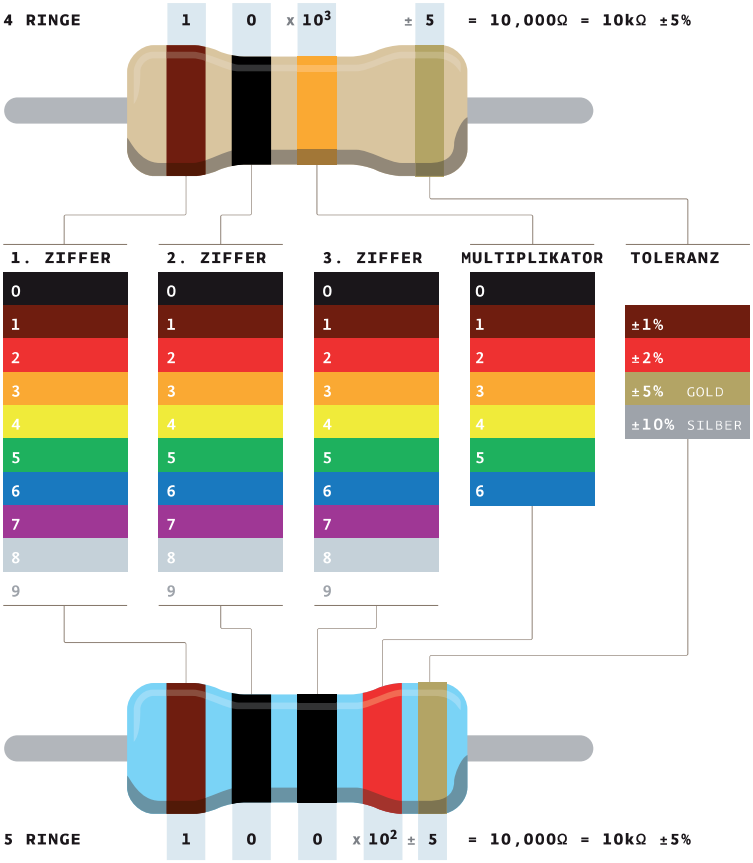
Wie würdest du die roten LEDs dazu bringen, beim Programmstart zu blinken? Wie kannst du eine größere oder komplexere Steuerkonsole mit mehr LEDs und Tastern für deine Weltraumabenteuer bauen?

Wenn du eine Bedienung für ein Gerät planst, überlege, was die Benutzer während der Verwendung erwarten. Soll es bei einem Tastendruck eine unmittelbare Reaktion geben? Oder sollte etwas Zeit vergehen zwischen der Benutzer-Aktion und dem, was der Genuino macht? Versetze dich in die Position eines anderen Benutzers während des Designs. Vergleiche die Erwartungen mit der Realität des Projektes.

*In diesem Projekt hast du dein erstes Genuino-Programm geschrieben. Darin kontrollierst du das Verhalten mehrerer LEDs mit einem Tasters. Du hast Variablen verwendet, die if-else-Anweisung benutzt und Funktionen zum Lesen einer Eingabe und zum Steuern einer Ausgabe angewendet.*

## WIDERSTÄNDE LESEN

Widerstandswerte werden mit farbigen Ringen codiert. Diese Methode wurde in den 1920er Jahren entwickelt. Damals war es zu schwierig, auf solche kleinen Objekte Zahlen zu schreiben. Jede Farbe steht für eine Zahl, wie in der Tabelle unten gezeigt. Jeder Widerstand hat entweder vier oder fünf Ringe. Bei den Typen mit vier Ringen stehen die ersten beiden Ringe für die ersten beiden Ziffern des Widerstandswertes, der dritte Ring für die Anzahl der folgenden Nullen (genauer: für die Zehnerpotenz). Der letzte Ring steht für die Toleranz: In unserem Beispiel besagt die Farbe Gold, dass der Widerstandswert 10 KOhm plus/minus 5% beträgt.




### WIDERSTÄNDE IM STARTERKIT

Du findest sie entweder in der Variante mit vier oder fünf Ringen.