



Experimente mit dem Super Starterkit

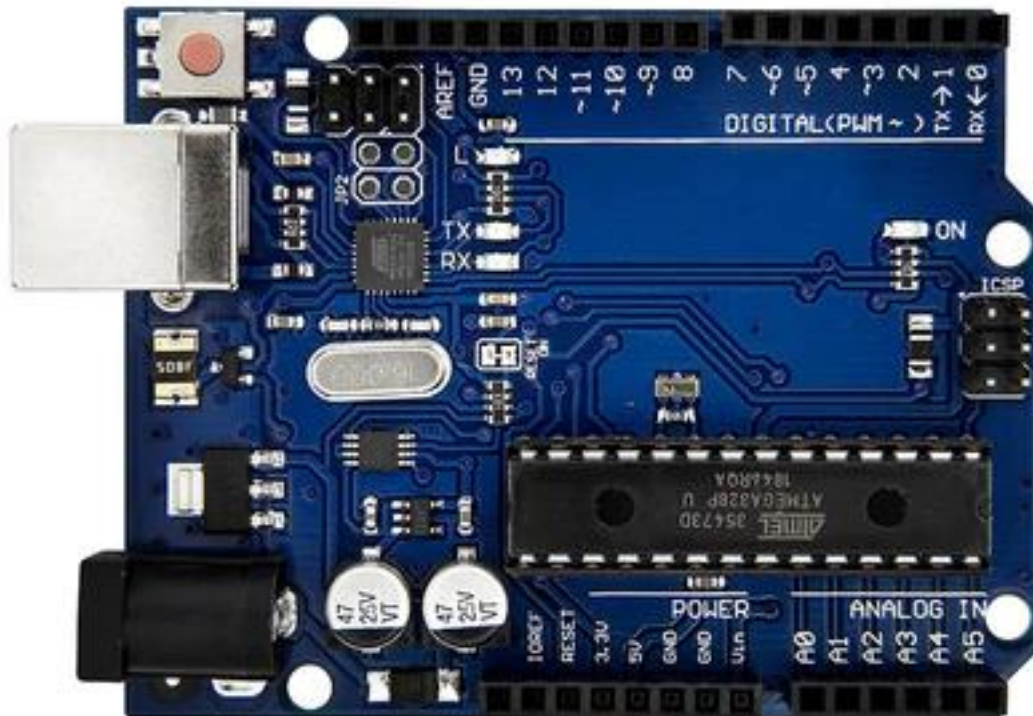
Teil 1 Arduino installieren und digitale Ausgänge

In diesem und folgenden Beiträgen werden Experimente vorgestellt, die allein mit den im Kit enthaltenen Bauteilen durchgeführt werden können. Die Beitragsreihe wird systematisch aufgebaut, sodass sie eine Einführung in die Nutzung eines Mikrocontrollers vermittelt. Außer dem Umgang mit einem Computer, sind keine Vorkenntnisse erforderlich.

Benötigte Hardware

Anzahl	Bauteil	Anmerkung
1	Super Starter Kit	
1	9 V Blockbatterie oder 9 V Netzteil	optional

Mikrocontroller



Kernstück des Sets ist das Board mit dem Mikrocontroller ATMEGA 328 P. Dieser Mikrocontroller entspricht einem kompletten Computersystem mit CPU, 32 k Byte Programmspeicher, 2 k Byte Datenspeicher und zwanzig Ein-/Ausgangsanschlüssen. Der 328P kann über eine serielle Schnittstelle programmiert werden. Der IO-Pin 0 wird als Eingang (RX) und der IO-Pin1 als Ausgang (TX) benutzt.

Auf dem im Set enthaltenen sogenannten Breakout-Board sind alle Ein-/Ausgänge auf Buchsenleisten herausgeführt. Über einen zweiten kleinen Mikrocontroller wird ein Gateway zwischen USB-Bus und serieller Schnittstelle realisiert. Über diesen Weg kann eine Kommunikation zwischen Computer und Mikrocontroller hergestellt werden. Außerdem sind auf dem Board ein Schwingquarz für die Takterzeugung und Spannungsregler für die Stromversorgung vorhanden.

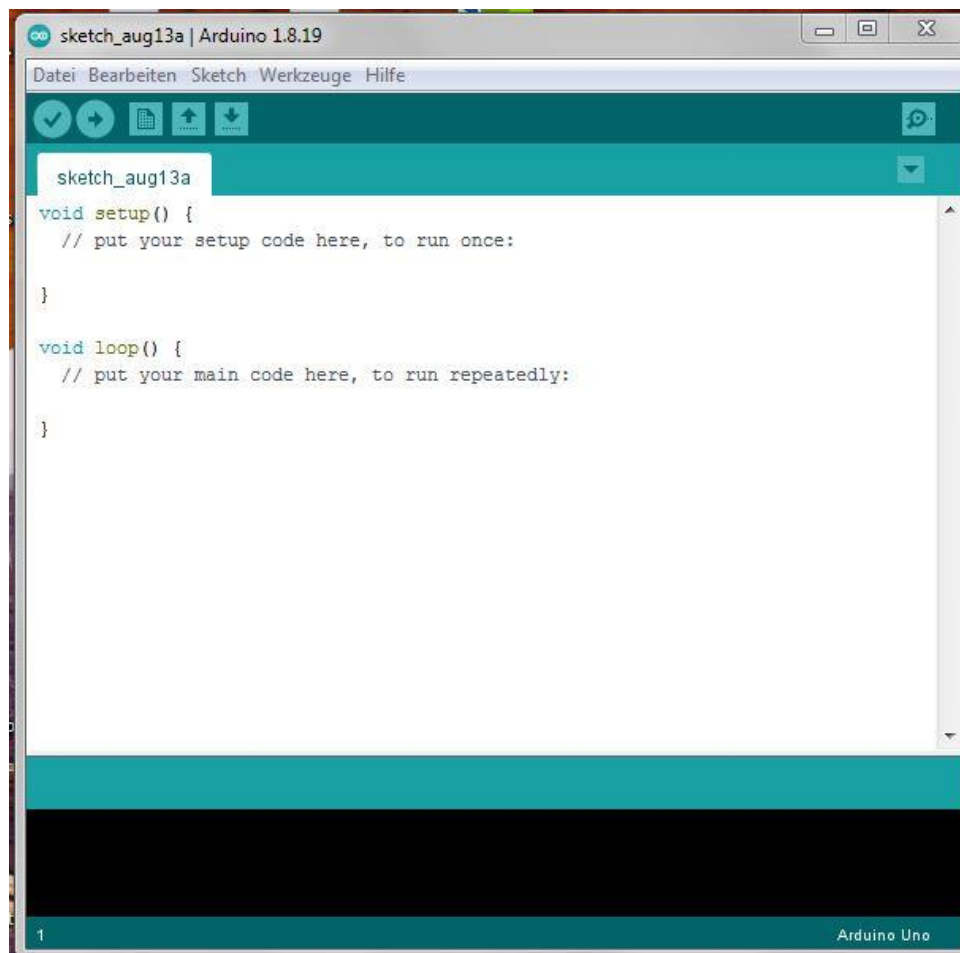
Es gibt zwei Möglichkeiten, um das Board mit Strom zu versorgen. Am einfachsten über den USB-Anschluss. Alternativ gibt es eine Buchse für eine externe Stromversorgung. Hier können Spannungen zwischen 7V und 12V verwendet werden. Typisch kommt meist 9V zum Einsatz. Der Stromversorgungseingang ist außerdem durch eine Diode gegen falsche Polung geschützt.

Entwicklungsumgebung

Um ein Programm für den Mikrocontroller zu erstellen, braucht man verschiedene Software-Werkzeuge. Wenn man das Programm nicht direkt im Maschinencode des ATMEGA erstellen möchte, braucht man einen Compiler für die gewünschte Programmiersprache. Außerdem benötigt man ein Programm, um den Übersetzten Code zum Mikrocontroller zu übertragen und dort in den Programmspeicher zu schreiben. Praktisch ist es, wenn ein Werkzeug alle diese Funktionen vereint. Man nennt das dann eine integrierte Entwicklungsumgebung kurz IDE (integrated development

environment). Die kostenlose Open-Source Software Arduino IDE ist solch ein Programm. Der Prozessor ATMEGA 328P wird direkt unterstützt. Die verwendete Programmiersprache ist C bzw. vereinfachtes objektorientiertes C++. Aber keine Angst, wenn man nicht in die Tiefe gehen will, kann man die Grundlagen einfach und schnell erlernen. Die IDE erledigt das meiste automatisch im Hintergrund, sodass man sich auf das eigentliche Programm konzentrieren kann. Die Software kann direkt auf <https://www.arduino.cc/en/software> heruntergeladen werden.

Wenn die IDE nach der Installation gestartet wird, zeigt sich das folgende Bild.



Es wird automatisch ein leeres Programm, hier Sketch genannt angelegt. Der Name des Sketches wird aus dem aktuellen Datum generiert. Den Namen kann man über „Datei->Speichern unter“ nach Wunsch ändern. Der vorgegebene Sketch enthält zwei Funktionen, die immer vorhanden sein müssen. Die Funktion `setup()` wird einmal beim Start des Mikrocontrollers aufgerufen, die Funktion `loop()` immer wieder in einer Endlosschleife.

Wenn nun der Mikrocontroller über das USB-Kabel mit dem Computer verbunden wird, sollte der USB-Treiber, der mit der Arduino Installation installiert wurde, aktiviert werden. Im Menüpunkt „Werkzeuge“ der IDE sollte nun im Punkt Port „COM# (Arduino Uno)“ erscheinen, wobei # für eine Portnummer steht, die auf jedem Computer anders sein kann. Dieser Port sollte ausgewählt werden. Unter dem Punkt „Board“ sollte „Arduino Uno“ stehen.

Nach dieser Vorbereitung steht einem ersten Sketch nichts mehr im Wege. Es soll über die serielle Schnittstelle das Wort „Hallo“ ausgegeben werden. Dazu müssen zwei Befehle in die Funktion `setup()` eingefügt werden.

```
void setup() {

  // put your setup code here, to run once:

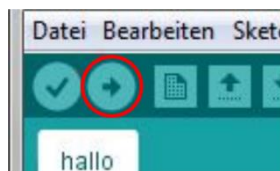
  Serial.begin(9600);

  Serial.println("Hallo!");

}
```

Der Befehl `Serial.begin(9600);` initialisiert die serielle Schnittstelle mit einer Geschwindigkeit von 9600 Baud (Bits/s). Mit dem zweiten Befehl `Serial.println("Hallo!");` wird das Wort „Hallo“ und ein Zeilenvorschub ausgegeben. Wird statt `println` nur `print` verwendet, erfolgt die Ausgabe ohne Zeilenvorschub. In C/C++ müssen alle Befehlszeilen mit einem Strichpunkt (Semikolon) abgeschlossen werden.

Damit die Ausgabe angezeigt werden kann, muss im Menüpunkt „Werkzeuge“ der Punkt „Serieller Monitor“ ausgewählt werden. Es öffnet sich ein extra Fenster für die Anzeige. In der Fußleiste sollte „9600 Baud“ als Geschwindigkeit angezeigt werden.



Klickt man nun auf den Pfeil oben links in der IDE, so wird der Sketch kompiliert und auf den Mikrocontroller übertragen. Nach der Übertragung wird ein Reset am Controller ausgelöst und damit das Programm gestartet. Im seriellen Monitor sollte das Wort „Hallo“ erscheinen.

Digitalausgänge

Alle 20 Ein-/Ausgänge können als digitale Ausgänge genutzt werden. Damit ein Anschluss als Ausgang genutzt werden kann, muss er mit dem Befehl `pinMode(13,OUTPUT);` (hier Pin 13) als Ausgang definiert werden. Der Ausgang kann zwei Werte annehmen. Mit dem Befehl `digitalWrite(13,1);` wird auf den Anschluss 13 der Wert 1 ausgegeben, das heißt der Pin geht auf 5V. Mit dem Befehl `digitalWrite(13,0);` wird auf den Anschluss 13 der Wert 0 ausgegeben, das heißt der Pin geht auf 0V.

Auf dem Board gibt es eine gelbe Leuchtdiode, die mit dem Pin 13 verbunden ist. Mit einem einfachen Sketch soll sie zum Blinken gebracht werden. Für den Sketch wird noch der Befehl `delay(1000);` benötigt. Dieser Befehl verzögert die Programmausführung um die angegebene Zahl in Millisekunden. In der Setup Funktion wird der Pin 13 als Ausgang definiert. Diesmal kommt auch die Loop Funktion zum Einsatz, um die LED ein- und auszuschalten.

```
void setup() {

  // put your setup code here, to run once:

  Serial.begin(9600);

  Serial.println("Start");

  pinMode(13,OUTPUT);

}
```

```

void loop() {

  // put your main code here, to run repeatedly:

  digitalWrite(13,1);

  delay(1000);

  digitalWrite(13,0);

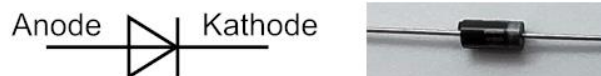
  delay(1000);

}

```

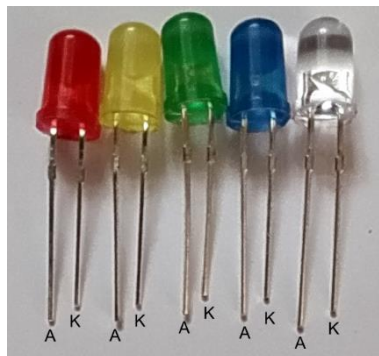
Diode und Leuchtdiode LED

Eine Diode oder auch Gleichrichter ist ein Bauelement, das die Stromleitung nur in eine Richtung zulässt. Legt man an der Anode eine positive Spannung in Bezug auf die Kathode an, so kann ein Strom fließen. Allerdings beginnt der Stromfluss nicht sofort, sondern erst, wenn die sogenannte Schwellspannung erreicht wird. Für Siliziumdioden ist die Schwellspannung ungefähr 0.7 Volt. Bei umgekehrter Polung fließt bis zur Durchbruchspannung kein Strom. Die Durchbruchspannung kann bei Dioden sehr hoch sein. Im Set sind zwei Dioden vom Typ 1N4007 enthalten. Die Schwellspannung ist 0.6 V, die Durchbruchspannung 1000 V und Strombelastbarkeit 1 A.



Die Abbildung zeigt das Schaltzeichen und ein Foto der Diode. Der Ring rechts kennzeichnet die Kathode

Eine Leuchtdiode verhält sich wie eine einfache Diode. Durch die Verwendung spezieller Halbleitermaterialien wird erreicht, dass oberhalb der Schwellspannung bei Stromfluss Lichtwellen ausgesendet werden. Die Schwellspannung ist deutlich höher und abhängig von der erzielten Leuchtfarbe. Der längere Anschlussdraht kennzeichnet die Anode. Im Set sind je fünf LEDs in den Farben rot, gelb, grün, blau und weiß enthalten.

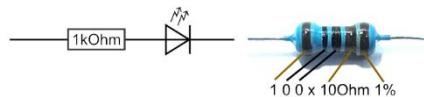


	rot	gelb	grün	blau	weiß
Schwellspannung	1.89 V	1.99 V	2.54 V	2.75 V	2.77 V

Damit eine Leuchtdiode an einer Spannungsquelle betrieben werden kann, ist ein Vorwiderstand notwendig, um den Strom zu begrenzen. Der Strom bestimmt außerdem die Helligkeit. Mit einem Vorwiderstand von 1000 Ohm erhält man für eine rote Leuchtdiode:

$$(U_{\text{out}} - U_{\text{Schwell}}) / R_v = (5 \text{ V} - 1.89 \text{ V}) / 1000 \text{ Ohm} = 3.11 \text{ mA}$$

Dieser Strom gibt eine ausreichende Helligkeit und überlastet den Ausgang des Mikrocontrollers nicht.

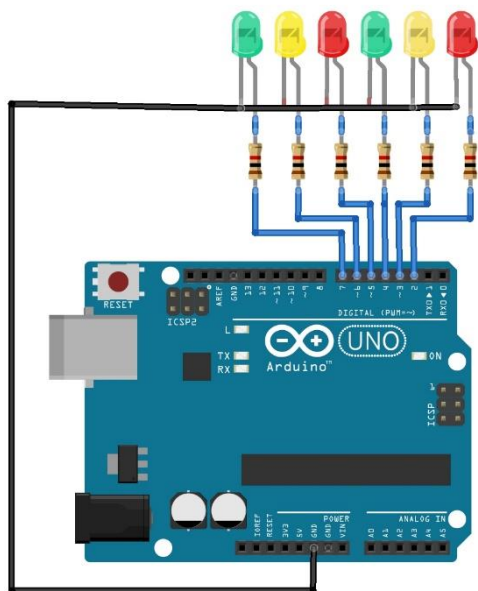


Sketch Verkehrsampel

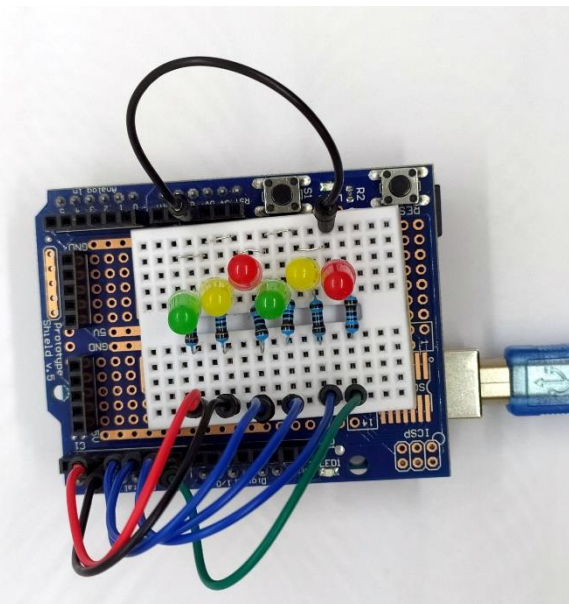
Als Anwendungsbeispiel soll der Mikrocontroller mit LEDs eine Verkehrsampel nachbilden. Die Signalabfolge soll sein:

Rot -> Rot/Gelb -> Grün -> Gelb -> Rot -> Rot/Gelb ->

Es sollen zwei Gruppen mit drei LEDs für beide Fahrrichtungen verwendet werden. Die Schaltung sieht dann so aus und kann am kleinen Steckbrett aufgebaut werden. Die LEDs werden mit den Digital-Ausgängen 2 bis 7 verbunden. Jede LED erhält einen Vorwiderstand mit 1 kOhm. Die Kathoden aller LEDs wird mit GND verbunden.



fritzing



Der Sketch enthält einige neue Kommandos. Mit `#define Name Konstante` kann man einer Konstanten einen Namen zuweisen. So wird das Programm lesbarer und wenn so eine Konstante mehrfach verwendet wird, kann ihr Wert einfach an einer Stelle geändert werden. Der Compiler ersetzt vor dem Kompilieren die Namen durch die entsprechenden Konstanten.

Damit der Zustand aller sechs LEDs in einem Byte gespeichert werden kann, wird jeder LED ein Bit zugeordnet. Setzt man 0B vor die Konstante, kann sie binär eingegeben werden.

```
#define GRUEN1 0B00000001
#define GELB1  0B00000010
#define ROT1   0B00000100
#define GRUEN2 0B00001000
#define GELB2  0B00010000
#define ROT2   0B00100000
```

Auch die Nummern der verwendeten Pins werden so definiert

```
#define PIN_GRUEN1 2
#define PIN_GELB1  3
#define PIN_ROT1   4
#define PIN_GRUEN2 5
#define PIN_GELB2  6
#define PIN_ROT2   7
```

Die Anzahl der Schritte gibt an, nach wie vielen Schritten sich das Muster wiederholt.

```
#define MAX_SCHRITTE 4
```

Nun folgt die Definition der globalen Variablen. Das sind Speicherplätze im RAM, auf die so lange zugegriffen werden kann, solange der Controller mit Strom versorgt wird. Global bedeutet, dass auf diese Variablen von überall aus zugegriffen werden kann. Für das Speichern der Signalabfolge werden zwei Listen von Werten benötigt. Eine Liste von Werten wird definiert, indem man zuerst den Datentyp angibt. Es folgt der Name für die Variable und in eckigen Klammern die Anzahl der Elemente. Damit ist die Variable fertig deklariert. Man kann ihr aber auch gleich Werte zuweisen. Die Zuweisung wird mit einem Gleichheitszeichen eingeleitet. Danach folgen in geschweiften Klammern die einzelnen Werte durch Komma getrennt.

Die erste Liste mit dem Namen *signale* definiert die Muster, indem die Bits für die LEDs, die leuchten sollten, gesetzt werden.

Die zweite Liste mit Namen *zeiten* enthält die Dauer für die einzelnen Schritte in Sekunden. Hier wird der Datentyp *int* für Werte zwischen -32767 und +32767 verwendet.

Es folgt eine einfache Variable vom Typ *byte*, also acht Bits in der immer der aktuelle Schritt gespeichert wird.

```
byte signal[MAX_SCHRITTE] = {
    GRUEN1 + ROT2,
    GELB1 + ROT2 + GELB2,
    ROT1 + GRUEN2,
    ROT1 + GELB1 + GELB2
};
```

```
int zeiten[MAX_SCHRITTE] = {6,2,10,2};
```

```
byte schritt = 0;
```

In der Setup-Funktion wird, wie im ersten Beispiel, die serielle Schnittstelle initialisiert und ein Text ausgegeben. Dann werden die Pins für die LEDs als Ausgang definiert und zum Schluss der Aktuelle Schritt auf 0 gesetzt.

```
void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.println("Start Ampel");
}
```

```

pinMode(PIN_GRUEN1, OUTPUT);
pinMode(PIN_GELB1, OUTPUT);
pinMode(PIN_ROT1, OUTPUT);
pinMode(PIN_GRUEN2, OUTPUT);
pinMode(PIN_GELB2, OUTPUT);
pinMode(PIN_ROT2, OUTPUT);
schritt = 0;
}

```

In der Funktion `loop` erfolgt das Schalten der LEDs. Zuerst wird eine lokale Variable `v` definiert, um das Muster für den aktuellen Schritt zu speichern. Lokal bedeutet, dass nur innerhalb dieser Funktion auf diese Variable zugegriffen werden kann. Ob eine LED leuchten soll oder nicht, findet man heraus, indem man das entsprechende Bit herausfiltert. Zum Filtern eignet sich die logische UND Verknüpfung.

Beispiel: Der zweite Schritt hat das Muster GELB1 + ROT2 + GELB2 oder binär 00110010. Eine UND Verknüpfung mit GRUEN1 = 00000001 liefert 00000000, da das entsprechende Bit nicht gesetzt ist. Eine UND Verknüpfung mit GELB1 = 00000010 liefert dagegen 00000010 oder dezimal 2.

Das Ergebnis der UND Verknüpfung kann mit der Funktion `digitalWrite(Pin, Wert);` direkt auf den Pin ausgegeben werden. Ist der Wert 0 geht der Pin auf 0 V und ist der Wert ungleich 0, geht der Pin auf 5 V und die LED leuchtet.

Nachdem alle Pins entsprechend dem Muster gesetzt wurden, werden zur Kontrolle die aktuellen Werte auf die serielle Schnittstelle ausgegeben. Dann wartet das Programm so lange, wie in der Liste `zeiten` angegeben wurde. Da die Werte in der Liste `zeiten` Sekunden sind, müssen diese mit 1000 multipliziert werden, da die Funktion `delay()` Millisekunden erwartet. Zu guter Letzt wird der Schrittzähler mit dem Befehl `schriftt++` um Eins erhöht. Wird die maximale Schrittzahl erreicht, so muss der Schrittzähler wieder auf 0 gesetzt werden.

```

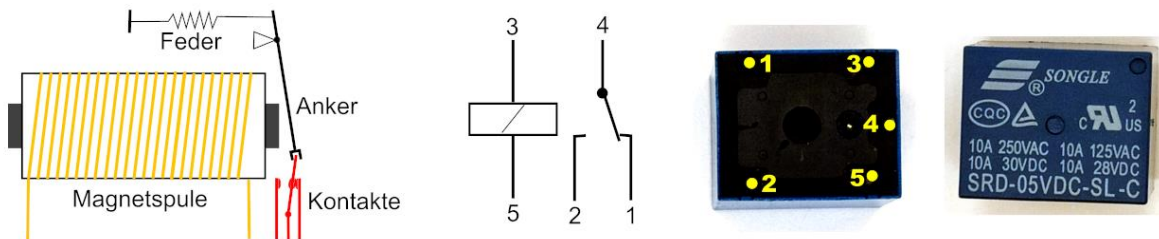
void loop() {
    byte v = signale[schritt];
    digitalWrite(PIN_GRUEN1, v & GRUEN1);
    digitalWrite(PIN_GELB1, v & GELB1);
    digitalWrite(PIN_ROT1, v & ROT1);
    digitalWrite(PIN_GRUEN2, v & GRUEN2);
    digitalWrite(PIN_GELB2, v & GELB2);
    digitalWrite(PIN_ROT2, v & ROT2);

    Serial.print("Schritt = ");
    Serial.print(schritt);
    Serial.print(" Dauer = ");
    Serial.print(zeiten[schritt]);
    Serial.println(" s");
    delay(zeiten[schritt]);
    schritt++;
    if (schritt == MAX_SCHRITTE) schritt = 0;
}

```

Relais

Ein Relais ist ein mechanischer Schalter, der über einen Elektromagneten gesteuert wird. Wenn Strom durch die Spule des Elektromagneten fließt, wird ein eiserner Anker angezogen, der den Schaltkontakt umschaltet. Die Abbildung zeigt den Aufbau, das Schaltzeichen und ein Bild des Relais von unten und oben.



Der Vorteil eines Relais ist die galvanische Trennung zwischen Steuerkreis und Lastkreis. So können damit von einem Mikrocontroller große Spannungen und Ströme geschaltet werden. Es ist außerdem gleichgültig, ob Gleich- oder Wechselstrom geschaltet werden soll. Das Relais im Starterkit wird mit 5V angesteuert. Im aktiven Zustand benötigt es einen Strom von 70 mA. Die Schaltkontakte können Ströme bis 10A bei Gleichspannungen bis zu 30 V oder Wechselspannungen bis zu 250 V schalten. Die Isolation zwischen Steuerkreis und Lastkreis verträgt eine Spannung von 1500 V.

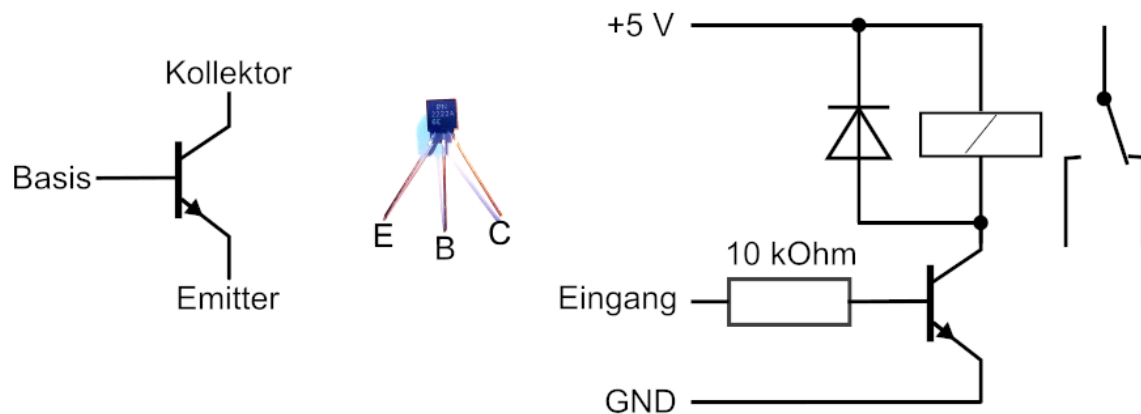
Transistor

Ein Transistor ist ein elektronisches Bauelement, das es ermöglicht, große Ausgangsströme mit kleinen Eingangsströmen zu steuern. Das Verhältnis Ausgangsstrom zu Eingangsstrom nennt man Stromverstärkung. Für die beiden Transistoren im Starter-Kit gilt eine Stromverstärkung von 300. Der Transistor hat drei Anschlüsse. Der Emitter ist für Eingang und Ausgang gemeinsam. Die Basis stellt den Eingang dar und verhält sich wie eine Diode. Wird die Schwellspannung von 0.7 V überschritten, fließt ein Eingangsstrom, der durch einen Vorwiderstand begrenzt werden soll. Dieser Eingangsstrom bewirkt, dass über die Strecke zwischen dem Kollektor, der den Ausgang darstellt, und dem Emitter abhängig vom Eingangsstrom ein proportionaler Ausgangsstrom fließt. Das geht so lange, bis die Spannung zwischen Kollektor und Emitter nur noch etwa 1V beträgt. Diesen Zustand nennt man Sättigung.

Da das Relais einen Strom von 70 mA zur Betätigung benötigt, kann es nicht direkt vom Mikrocontroller angesteuert werden. Hier kann nun der Transistor eingesetzt werden. Für einen Ausgangsstrom von 70 mA benötigt man einen Eingangsstrom von $70 / 300 = 0.23 \text{ mA}$. Der Mikrocontroller liefert am Ausgang eine Spannung von 5 V. Abzüglich der Schwellspannung von 0.7 V liegen dann am Vorwiderstand 4.3 V. Somit ergibt sich für den Vorwiderstand $4.3 \text{ V} / 0.23 \text{ mA} = 18.4 \text{ k}\Omega$. Im Starterkit gibt es Widerstände mit 10 k Ω . Mit dem 10 k Ω Widerstand ergibt sich ein Eingangsstrom von 0.43 mA. Das bedeutet, dass das Relais sicher genug Strom für ein sicheres Schalten erhält.

Da die Magnetspule eine Induktivität darstellt, versucht sie beim Ausschalten den Strom beizubehalten. Da dies wegen des unterbrochenen Stromkreises nicht möglich ist, entsteht an der Spule eine hohe Spannungsspitze, die den Transistor zerstören würde. Daher wird eine Diode parallel zur Spule geschaltet, die der Spule die Möglichkeit gibt, den Strom abklingen zu lassen. Dadurch

werden die Spannungsspitzen vermieden. Die Abbildung zeigt das Schaltzeichen für einen Transistor, ein Bild des Transistors aus dem Starterkit und die fertige Ansteuerschaltung für ein Relais.



Die zwei Transistoren im Starterkit sind vom Typ PN2222A und haben folgende Kenndaten:

Stromverstärkung 300,
Basis Schwellspannung 0.66 V,
Kollektor – Emitter Sperrspannung maximal 30 V,
Kollektorstrom maximal 600 mA

Der Eingang kann mit einem Pin des Mikrocontrollers verbunden werden. Die Ansteuerung erfolgt wie bei der Leuchtdiode mit dem Befehl `digitalWrite(pin, wert)`.