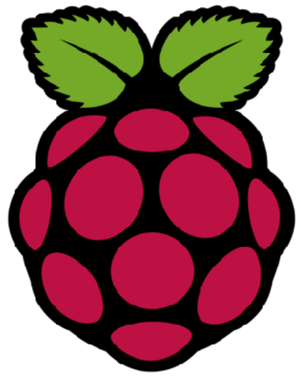
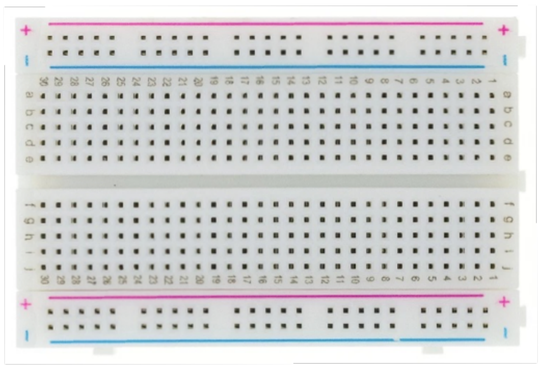
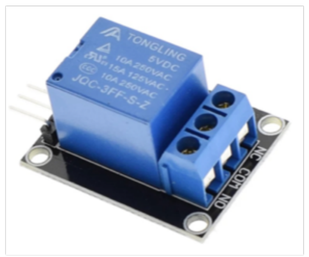
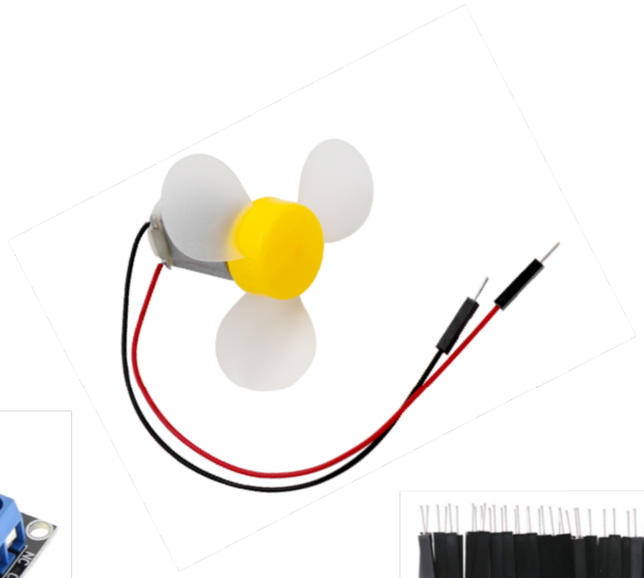


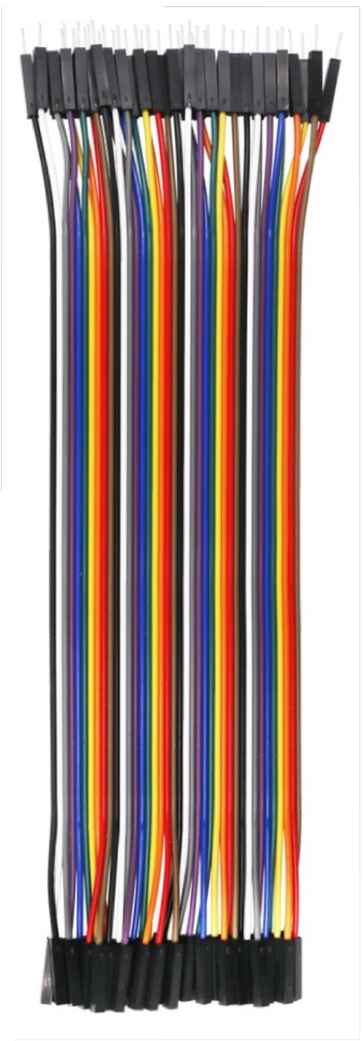
+

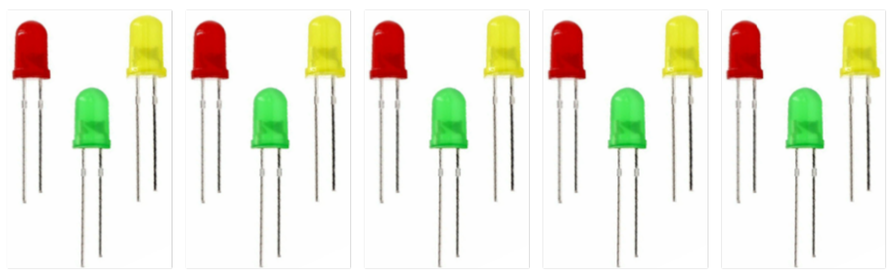
python™ + GPIO zero











Elektronik-Guide Raspberry Pi Edition

Impressum

Elektronik-Guide Raspberry Pi Edition

Version: 2022-12-16

Herausgeber:
Patrick Schnabel
Droste-Hülshoff-Str. 22/4
71642 Ludwigsburg
Deutschland

USt-ID-Nr.: DE207734730
WEEE-Reg.-Nr.: DE80632679

https://www.elektronik-kompendium.de/

CE

Dieses Elektronik-Set wurde nach den geltenden europäischen Richtlinien entwickelt und hergestellt. Der bestimmungsgemäße Gebrauch aller Bauteile ist in dieser Anleitung beschrieben. Der Nutzer ist für den bestimmungsgemäßen Gebrauch und Einhaltung der geltenden Regeln verantwortlich. Bauen Sie die Schaltungen deshalb nur so auf, wie es in dieser Anleitung beschrieben ist. Das Elektronik-Set darf nur zusammen mit dieser Anleitung weitergegeben werden.

Das Symbol mit der durchkreuzten Mülltonne bedeutet, dass dieses Produkt nicht mit dem Hausmüll entsorgt, sondern als Elektroschrott dem Recycling zugeführt werden muss. Mit dem Kauf dieses Produkts wurden die Gebühren für die Entsorgung entrichtet. Die nächstgelegene kostenlose Annahmestelle für Elektroschrott erfahren Sie von Ihrer regional zuständigen Abfallwirtschaft.

Vorwort (1)

Elektronik, einfach und leicht verständlich. Das ist mein Motto. Auch zusammen mit dem Raspberry Pi. Aber, Elektronik und Raspberry Pi ist eine schwierige Ehe. So dachte ich am Anfang. Die Schwierigkeit liegt nicht in der Elektronik, die am Raspberry Pi in ein enges Korsett gezwängt wird. Nein, es ist die Kombination aus Elektronik und Software-Programmierung. Denn wenn etwas nicht funktioniert, wo liegt dann der Fehler? In der Verschaltung der Bauteile oder im Programmcode? Bei dieser doppelten Komplexität wird aus Lust schnell Frust. Das weiß ich aus Erfahrung.

Wir brauchen also möglichst einfache Schaltungen und ganz kurze und leicht lesbare Quelltexte. Nach Möglichkeit solltest Du nicht zu viel falsch machen können. Außer die üblichen Bedienungsfehler. Die Raspberry-Pi-Community hat gegen die Komplexität eine Lösung gefunden: GPIO Zero.

GPIO Zero ist eine Python-Bibliothek zum Programmieren von elektronischen Bauteilen. Oder, wie es heute genannt wird, Physical Computing oder auch Hardware-nahes Programmieren. GPIO Zero reduziert die Komplexität auf wenige Zeilen Programmcode.

Auf der Programmierseite ist GPIO Zero zusammen mit Python die ultimative Lösung für Einsteiger in die Hardware-nahe Programmierung. Aber auch Fortgeschrittene, die bisher alles mit „RPi.GPIO“ gemacht haben, lohnt sich ein Blick darauf. Ich finde, das Programmieren von GPIOs ist dadurch viel einfacher. Du kannst Dich mehr auf die eigentliche Funktionalität Deines Programms konzentrieren.

Jetzt brauchst Du nur noch ein dazu passendes Bauteile-Sortiment zum Experimentieren. Das gibt es natürlich auch. Allerdings muss man die im Ausland bestellen. Wenn sie denn mal verfügbar wären. Mir ist schon klar, dass diese billigen Elektronik-Sets mit

Vorwort (2)

einer handvoll Bauteile kein großes Geschäft bedeuten. Das ist für die großen und bekannten Raspberry-Pi-Shops eher so ein Mitnahme-Produkt. Entsprechend stiefmütterlich sind diese Sets ausgestattet und nur gelegentlich verfügbar.

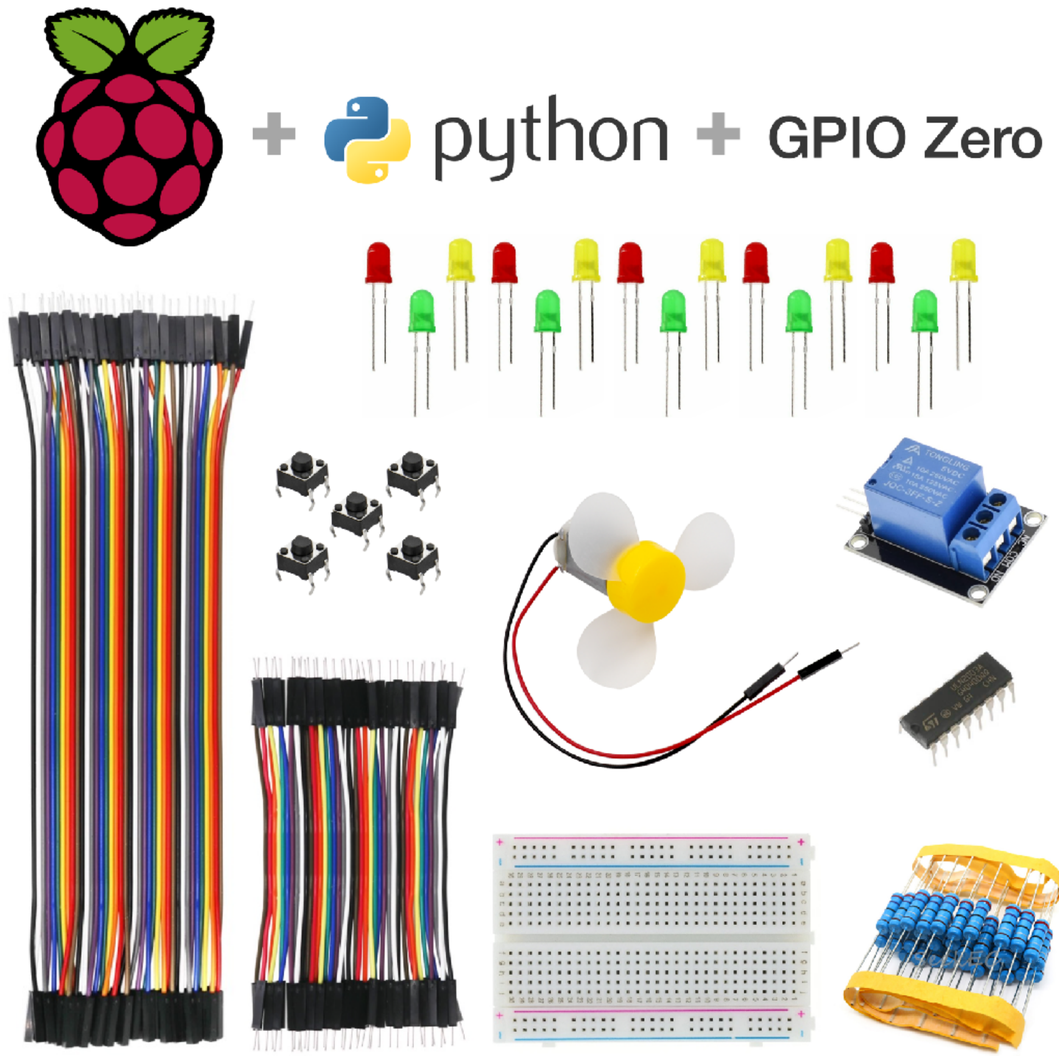
Das ist wirklich schade. Der Python-Bibliothek GPIO Zero wird das nicht gerecht. Aus meiner Sicht muss das nicht sein. Deshalb garantiere ich die dauerhafte Verfügbarkeit des Elektronik-Sets „Raspberry Pi Edition“.

Für diesen Elektronik-Guide „Raspberry Pi Edition“ habe ich mir die offizielle englischsprachige Dokumentation von GPIO Zero zur Brust genommen und alle Beispiele ausprobiert. Ich habe mir dann überlegt, welche Bauteile wirklich notwendig sind, um nicht alle, aber möglichst viele sinnvolle Experimente durchzuführen. Die meisten Aufbauten und Programmcodes weichen deshalb vom Original ab, damit man mehrere Programmcodes nacheinander ausprobieren kann, ohne Änderungen am Aufbau vornehmen zu müssen.

Herausgekommen ist das „Elektronik-Set Raspberry Pi Edition“ mit Anleitung und Bauteilen zum Experimentieren und Programmieren mit Python und GPIO Zero. Dieser Elektronik-Guide soll dabei kein Ersatz für die offizielle Dokumentation sein, sondern die Schwelle zum Einstieg noch ein wenig tiefer legen.

Ich finde, so macht Experimentieren mit dem Raspberry Pi und Elektronik viel mehr Spaß.

Elektronik-Set Raspberry Pi Edition



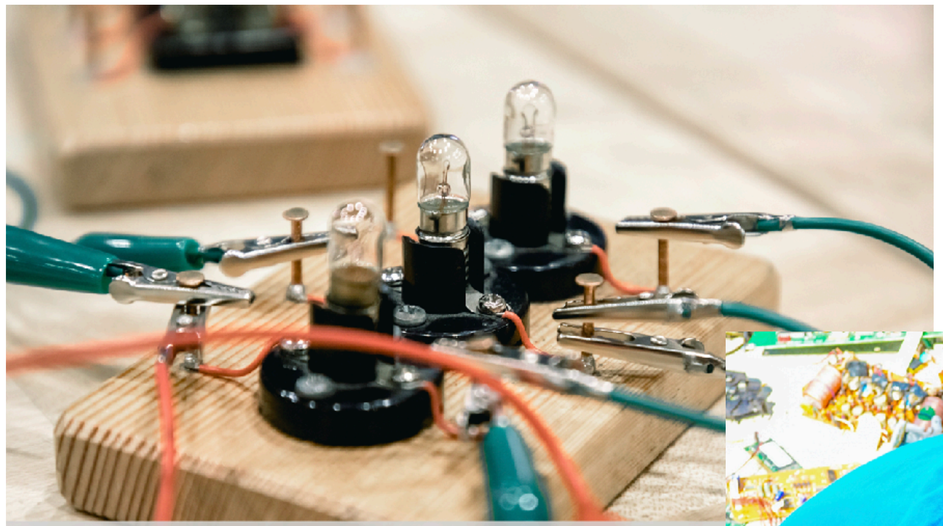
Das Elektronik-Set Raspberry Pi Edition enthält viele elektronische Bauteile, um Hardware-nahes Programmieren zu lernen, Steuerungen selber zu programmieren und ohne Lötkolben zu experimentieren.

Eine strukturierte Einführung berücksichtigt die Besonderheiten von Elektronik und Programmierung ohne Vorkenntnisse.

- **Hardware-nahes Programmieren mit Python und GPIO Zero ohne Vorkenntnisse.**
- **Grundlagen lernen, um Steuerungen selber zu programmieren.**
- **Ohne Lötkolben experimentieren. Bauteile einfach stecken.**

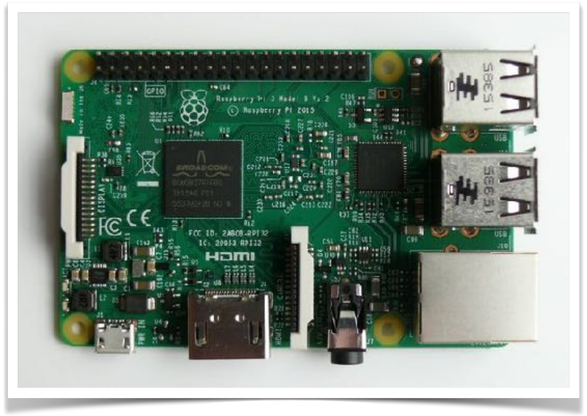
https://www.elektronik-kompendium.de/shop/elektronik-set/raspberry-pi-edition

Inhaltsverzeichnis



Einführung und Grundlagen

Raspberry Pi: Was ist das?



Der Raspberry Pi ist ein Mini-Computer, der ursprünglich für Schüler und Studenten gedacht war. Grundsätzlich ist er auch für Bastler, Maker und Hobby-Elektroniker interessant. Ganz nebenbei kann man damit viel über die grundsätzliche Funktionsweise von Computern, Programmieren und auch Elektronik lernen.

Modelle

Den Raspberry Pi gibt es in drei verschiedenen Größen. Als kleinen und sparsamen Raspberry Pi Zero. Als ebenfalls sparsam und leistungsfähigeren Raspberry Pi 3 A+. Und das Topmodell Raspberry Pi 4 B mit vier schnellen Rechenkernen und wahlweise 1, 2 oder 4 GByte Arbeitsspeicher.

Typische Anwendungen

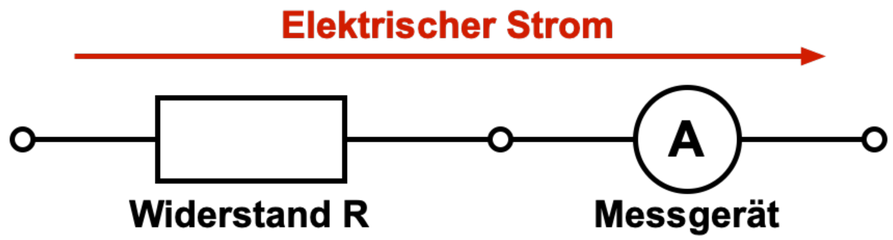
Die Rechenleistung des System-on-Chip (Prozessor) ist nicht besonders groß, weshalb man mit dem Raspberry Pi beim Experimentieren immer wieder an dessen Leistungsgrenzen stößt. Für viele Anwendungen reicht die Leistungsfähigkeit vollkommen aus.

- Mini-PC
- Steuerungs-Computer
- Multimedia-Center
- Netzwerk-Speicher

Spannung und Strom

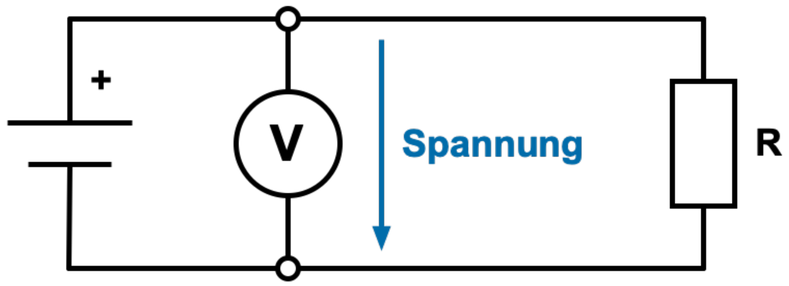
Was ist der elektrische Strom?

- Der elektrische Strom ist die Übertragung elektrischer Energie durch die Bewegung freier Ladungsträger (Elektronen oder Ionen).
- Die technische Stromrichtung wird in Schaltungen mit einem roten Pfeil von Plus (+) nach Minus (-) angegeben.
- Das Formelzeichen des elektrischen Stroms bzw. der elektrischen Stromstärke ist das große „I“.
- Die gesetzliche Grundeinheit des elektrischen Stroms ist Ampere (A). Kleinere Ströme werden in Milliampere (mA) und größere in Kiloampere (kA) angegeben.

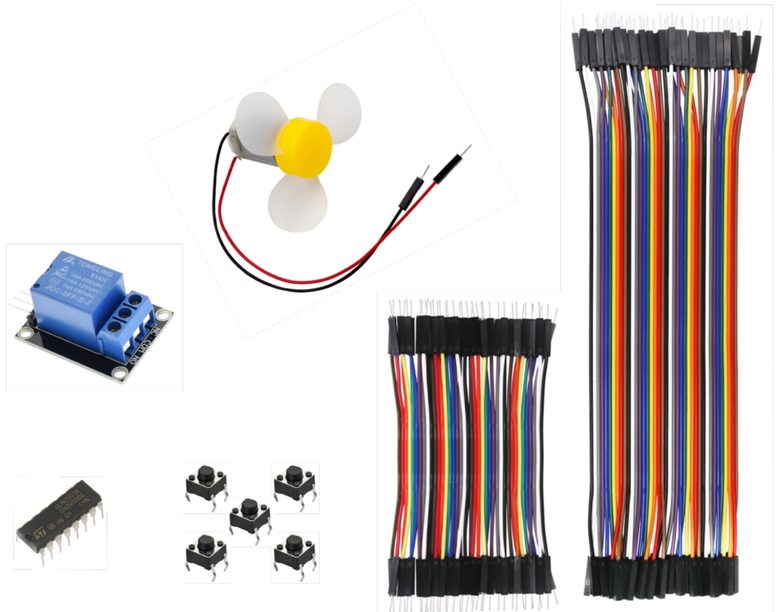


Was ist die elektrische Spannung?

- Die elektrische Spannung ist die Kraft auf freie Elektronen und die Ursache des elektrischen Stroms. Die elektrische Spannung entsteht durch einen Ladungsunterschied.
- Der Ladungsunterschied wird oft als blauer Pfeil von Plus (+) nach Minus (-) angegeben.
- Das Formelzeichen der elektrischen Spannung ist das große „U“.
- Die gesetzliche Grundeinheit der elektrischen Spannung ist Volt (V). Kleinere Spannungen werden in Millivolt (mV) und größere in Kilovolt (kV) oder Megavolt (MV) angegeben.



Elektronik mit dem Raspberry Pi



Elektronik mit dem Raspberry Pi ermöglicht Dank schaltbarer digitaler Ein- und Ausgänge vielfältige Anwendung im Bereich der Steuerung von elektronischen Bauteilen.

„Digital“ bedeutet, dass die Ein- und Ausgänge der binären Logik folgen und nur zwei Zustände annehmen bzw. unterscheiden können. Das heißt aber auch, analoge Signale können an den Eingängen nicht ausgewertet werden. Typische Mikrocontroller haben hier mehr Möglichkeiten.

Einschränkungen

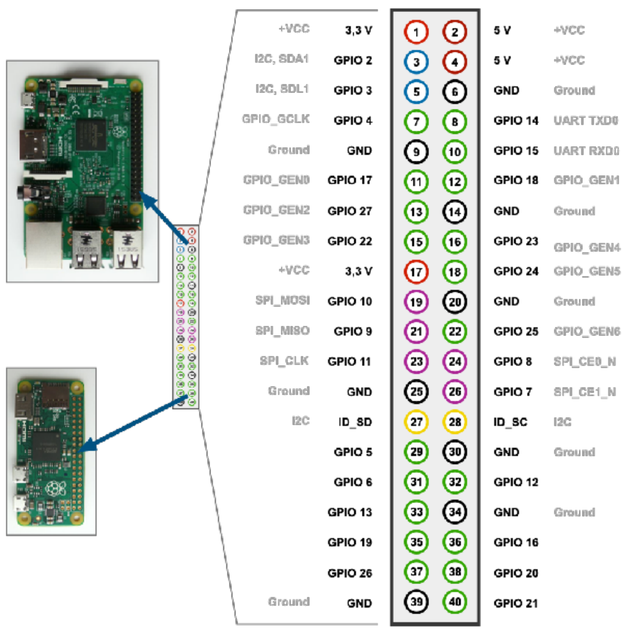
- Nur digitale Eingänge und Ausgänge
- Keine analogen Eingänge und Ausgänge, nur mit ADC-Chip möglich
- Keine Ströme schaltbar
- Defektanfälligkeit bei fehlerhafter Beschaltung

Wann ist Elektronik mit einem Raspberry Pi sinnvoll?

Ein Raspberry Pi für Physical Computing oder Hardware-nahes Programmieren ist dann sinnvoll, wenn die Hardware einem vollwertiger Computer ähneln muss. Zum Beispiel zur Nutzung in einer Netzwerk-Umgebungen oder die Installation von zusätzlicher Software.

Die Stärken des Raspberry Pi liegen klar in der Computer-Architektur, die die Ausführung von Open-Source-Software für Linux ermöglicht.

GPIO - General Purpose Input Output



Grundsätzlich, ein Raspberry Pi wird mit 5 Volt per USB-Netzteil gespeist. Allerdings arbeitet der System-on-Chip auf Basis von 3,3 Volt. Das bedeutet, an einem GPIO muss das Signal eingangsseitig auf 3,3 Volt begrenzt werden und ausgangsseitig ist das Signal auf 3,3 Volt begrenzt.

Desweiteren verträgt ein GPIO nur ein paar Milliampere (mA) Strom. Das bedeutet, über einen GPIO-Pin kann man keine externen Schaltungen und Bauteile betreiben. Die müssen eine eigene Stromversorgung haben.

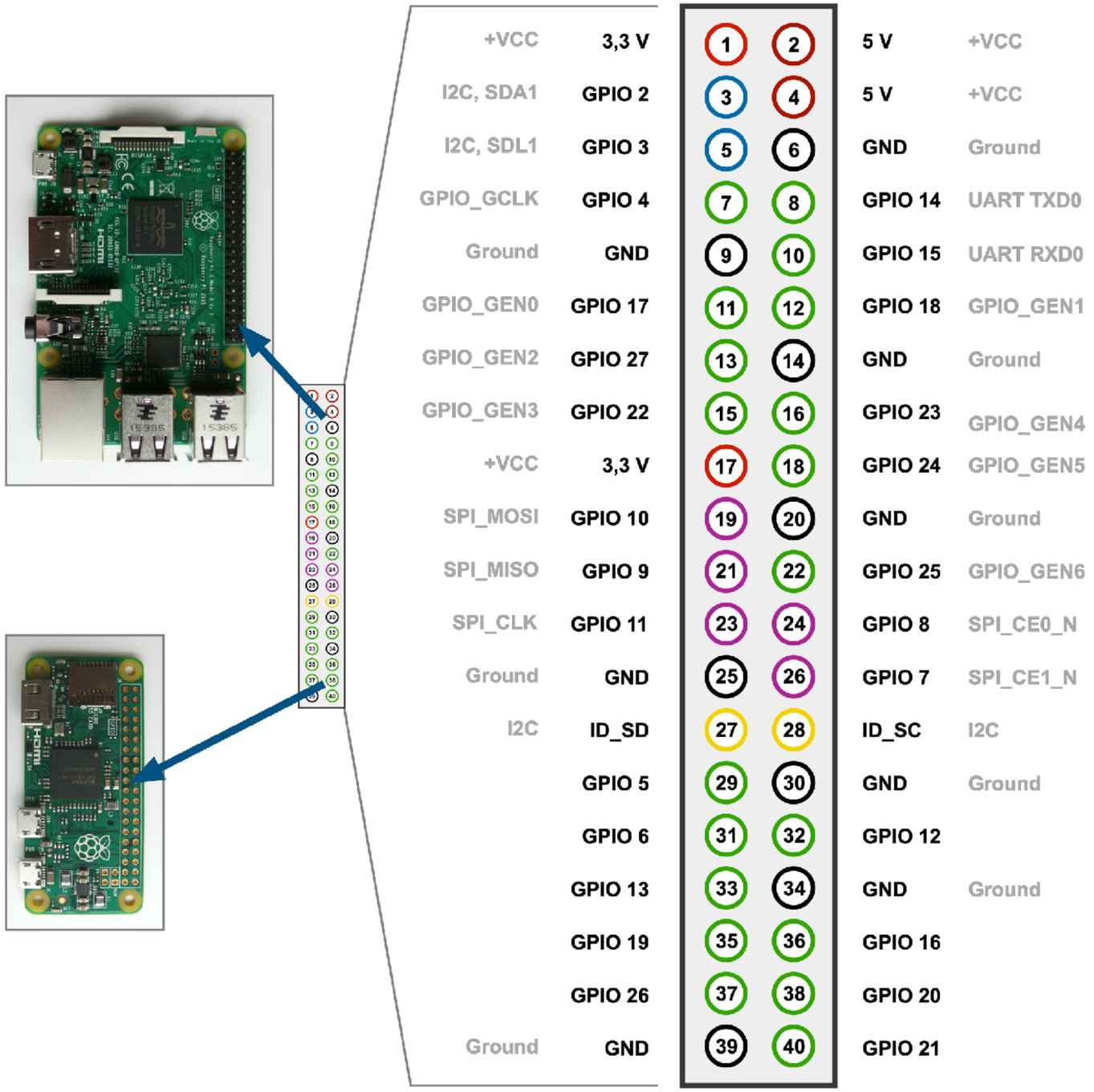
Insbesondere bei Relais und Motoren muss man sich um eine sichere Ansteuerung kümmern.

General Purpose Input Output, kurz GPIO, bezeichnet programmierbare Ein- und Ausgänge, die man für unterschiedliche Zwecke nutzen kann. GPIOs werden auf Platinen als Lötpunkte oder Pins in Form einer Stiftleiste herausgeführt und dienen als Schnittstelle zu anderen Systemen, Schaltungen oder Bauteilen, um diese über einen Raspberry Pi zu steuern. Dabei kann der Raspberry Pi bei entsprechender Programmierung digitale Signale von außen annehmen (Input) oder Signale nach außen abgeben (Output).

Viele der GPIOs erfüllen je nach Einstellung und Programmierung unterschiedliche Funktionen. Neben den typischen GPIO-Ein- und Ausgängen finden sich aber auch Pins mit der Doppelfunktion für I2C, SPI und eine serielle Schnittstelle.

Die GPIOs sind oftmals nur als Lötpunkte oder, beim Raspberry Pi vom Modell abhängig, als zweireihige Stiftleiste herausgeführt. Je nach Modell weist ein Raspberry Pi eine Stiftleiste mit 26 oder 40 Pins auf, wovon allerdings nicht alle zur Ein- oder Ausgabe dienen. Einige Pins führen ein festgelegtes +5V-, +3,3V- oder 0V-Potential.

GPIO-Belegung



GPIOs mit Python und GPIO Zero programmieren

Python ist eine universelle Programmiersprache. Sie gilt als einfach zu erlernende Sprache, da sie über eine klare und übersichtliche Syntax und einer guten Lesbarkeit verfügt. Des Weiteren ist Python in der Informatik weit verbreitet, so dass man sie in Technik-nahen Ausbildungen und Berufen immer wieder findet.

GPIO Zero ist eine Python-Bibliothek. Sie stellt eine Software-Schnittstelle für Physical Computing bereit, um LEDs, Taster, LED-Reihen, 7-Segment-Anzeigen, Analog-Digital-Wandler und vieles mehr über die GPIOs in Python zu programmieren und zu steuern.

Es gibt viele verschiedene Möglichkeiten um die GPIOs zu steuern und zu programmieren. Welche man verwendet hängt von den Anforderungen ab, oder einfach auch von den eigenen Fähigkeiten sich in bestimmte Programmiersprachen und Bibliotheken einzuarbeiten.

Zusammen sind Python und GPIO Zero für Anfänger und Einsteiger einfach unschlagbar. Und auch Fortgeschrittene können von der Einfachheit profitieren. Alltägliche Funktionen, die mit vielen Zeilen Programmcode realisiert werden müssten, sind in GPIO Zero implementiert und können mit einem einzigen Kommando aufgerufen werden.

Entwicklungsumgebung zum Programmieren

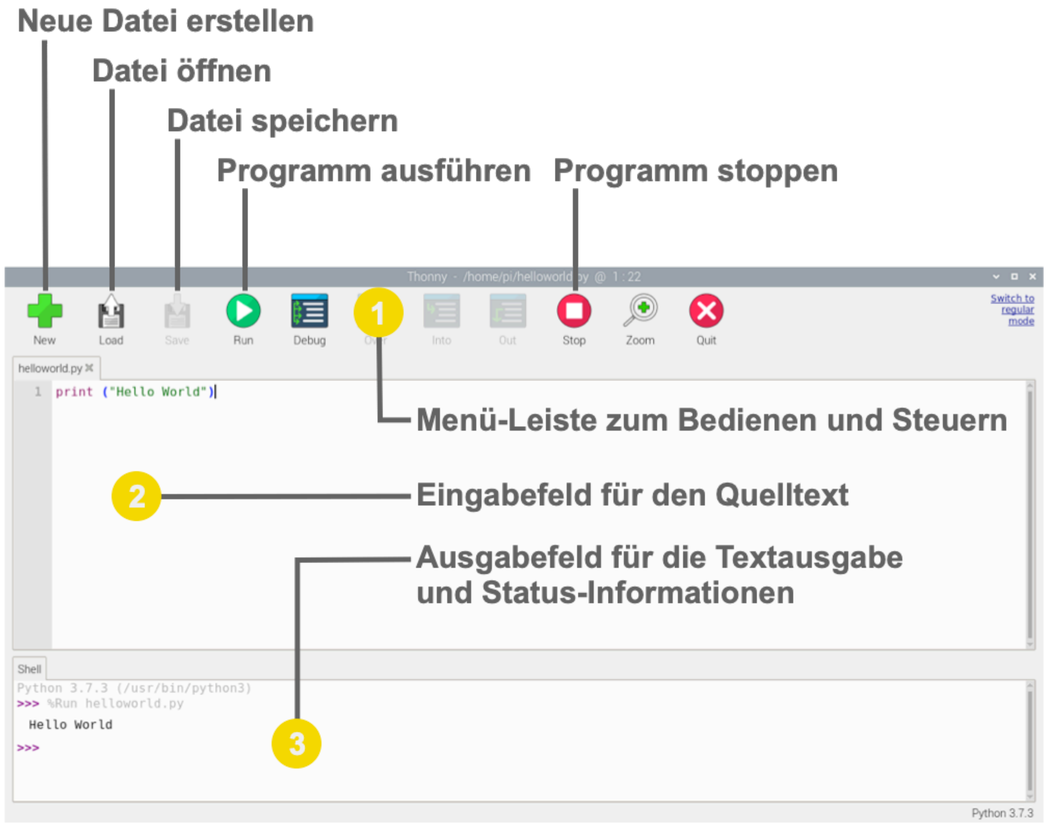
Was ist eine Entwicklungsumgebung?

Mit Entwicklungsumgebung ist meist ein Programm gemeint, in dem der Programmcode oder Quelltext für ein anderes Programm geschrieben wird. Also das Programm, mit dem programmiert wird. Im einfachsten Fall ist das ein einfacher Text-Editor. Programmierer verwenden gerne spezielle Text-Editoren die über Syntax-Highlightning verfügen. Das heißt, die Grammatik der Programmiersprache erkennen und zur besseren Lesbarkeit bestimmte Befehle, Kommandos, Optionen, Parameter usw. farblich unterschiedlich darstellen. Hilfreich sind Editoren, die auch noch Eingabefehler erkennen und kennzeichnen können. Desweiteren vereinfacht es die Software-Entwicklung, wenn man direkt im Editor das geschriebene Programm starten und stoppen kann.

Zu einer vollständigen Entwicklungsumgebung gehört meist eine bestimmte Hardware und zusätzliche Software, was aber davon abhängig ist, in welcher Programmiersprache geschrieben werden soll und was genau programmiert wird. Beispielsweise, ob eine bestimmte Hardware berücksichtigt werden muss.

- Hardware: Raspberry Pi mit Zubehör (Maus, Tastatur, Bildschirm)
- Elektronik: verschiedene Bauteile, Steckbrett und Verbindungskabel
- Betriebssystem: Raspberry Pi OS Desktop auf SD-Speicherkarte
- Programmiersprache: Python Version 3 mit GPIO Zero (in der Regel vorinstalliert)
- Editor: Thonny Python IDE (in der Regel vorinstalliert)
- Internet: von Vorteil, aber nicht zwingend erforderlich

Programmieren mit der Thonny Python IDE



Wenn Du mit der Desktop-Version von Raspberry Pi OS arbeitest, dann empfiehlt es sich mit der Thonny Python IDE zu programmieren.

Die Thonny Python IDE erreicht man im Programm-Menü über das Untermenü „Entwicklung“.

Die Thonny Python IDE verfügt über alle erforderlichen Funktionen und Bedienelemente, um effektiv mit Python und GPIO Zero programmieren zu können

Aufteilung der Entwicklungsumgebung:

1. Menü-Leiste zum Bedienen und Steuern
2. Eingabefeld für den Quelltext
3. Ausgabefeld für die Textausgabe und Status-Informationen

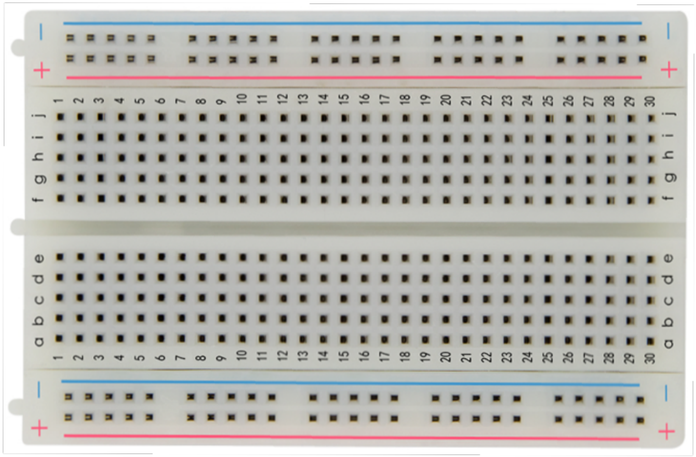
Wichtige Funktionen der Menü-Leiste:

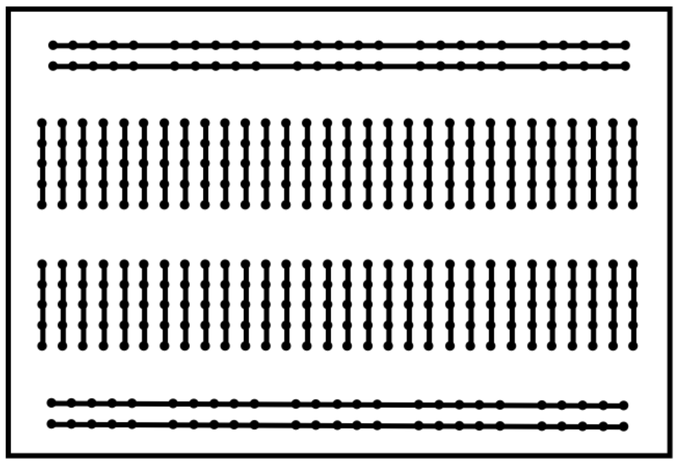
- Neue Datei erstellen (New)
- Datei öffnen (Load)
- Datei speichern (Save)
- Programm ausführen (Run)
- Programm stoppen (Stop)

Weitere Funktionen sind selbsterklärend oder erfüllen eine Spezialfunktion.

Bauteile

Steckbrett





Die durchgezogenen Linien kennzeichnen die Steckkontakte, die eine elektrische und damit leitende Verbindung aufweisen.

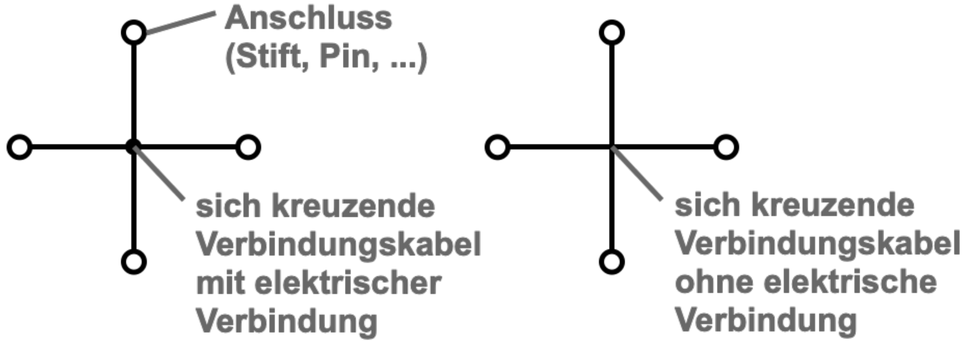
- lötfreies Experimentieren
- einfach zu verwenden und miteinander kombinierbar
- verschiedene Größen erhältlich

Eine Steckbrett ist eine Experimentier-Platine, die auch „Steckbrett“ oder „Steckboard“ genannt wird. Auf dem Steckbrett lassen sich elektronische Bauteile durch Stecken mechanisch Befestigen und zu Schaltungen elektrisch verbinden.

Das Steckbrett wird im Englischen oft „Solderless Breadboard“ genannt, womit angedeutet wird, dass das Verbinden mit dem Steckbrett lötfrei erfolgt. Das Steckbrett hat 400 Kontakte im 2,54-mm- bzw. 1-Zoll-Raster, die reihenweise miteinander verbunden sind. In der Mitte befindet sich ein breiter Steg, damit die Anschlüsse von integrierten Schaltungen (ICs) in DIL-Bauform voneinander getrennt sind.

Über die Rot und Blau markierten Seitenleisten erfolgt in der Regel die Stromversorgung für die gesteckten Schaltungen.

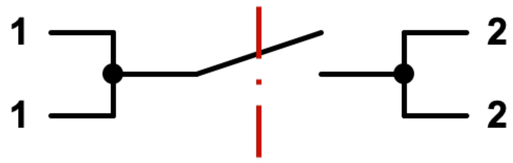
Steckverbindungskabel



- elektrische Verbindungen zwischen Bauelementen, Schaltungen und Geräten

Steckverbindungskabel oder auch nur Verbindungskabel dienen zum Trennen und Verbinden von Bauelementen, Schaltungen und Geräten.
Bei elektrischen Steckverbindungen unterscheidet man den männlichen Teil einer Steckverbindung (mit nach außen weisenden Kontaktstiften) und den weiblichen Teil (mit nach innen weisenden Kontaktöffnungen). Den männlichen Teil bezeichnet man als Stecker, wenn er sich an einem Kabelende befindet. Den weiblichen Teil bezeichnet man richtigerweise als Kupplung, wenn er sich an einem Kabelende befindet. Manchmal sagt man auch Buchse dazu.

Taster



Die im Bild mit 1 und 2 gekennzeichneten Anschlüsse können je nach Bauform an unterschiedlichen Positionen liegen. Welche Anschlüsse zusammengehören, muss man mit einem Durchgangsprüfer oder Widerstandsmessgerät herausfinden.

- Stromkreis schließen
- Stromkreis unterbrechen
- manuelles Ein- und Ausschalten

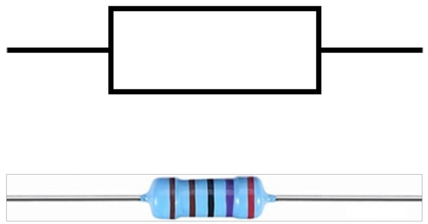
Durch einen Taster kann ein Stromkreis geschlossen und unterbrochen werden. Damit steuert man, ob im Stromkreis ein Strom fließt oder nicht.

Bei dem dargestellten Taster handelt es sich um einen 2-poligen Taster mit 4 Anschlüssen. Je zwei Anschlüsse stellen einen Kontakt des Tasters dar.

Bei einem Taster ist der geöffnete Zustand der Normalzustand. Der Kontakt wird nur durch Drücken des Tasters geschlossen. Und der Kontakt bleibt nur solange geschlossen, wie der Taster gedrückt bleibt. Lässt man den Taster los, dann öffnet sich der Kontakt wieder.

Wenn man einen Stromkreis richtig ein- und ausschalten will, dann sollte man dafür einen Schalter verwenden, der nach der Betätigung in seiner Position bleibt. Ein Taster schaltet sich automatisch in seinen Normalzustand zurück.

Widerstand



- Begrenzen des elektrischen Stroms
- Begrenzen der elektrischen Spannung
- Einstellen von Strom und Spannung

Mit einem Widerstand kann man Strom und Spannung in einer Schaltung begrenzen und bestimmte Spannungs- und Stromwerte an bestimmten Punkten in einer Schaltung einstellen.

Ein Widerstand ist zwar ungepolt, doch auch er kann kaputt gehen, wenn er zu viel Spannung oder zu viel Strom ausgesetzt wird. Die relevante Größe ist die elektrische Leistung in Watt (W) oder Milliwatt (mW). Das ist mathematisch ein Produkt aus Spannung und Strom. In der Regel verwendet man Widerstände mit maximal 250 mW bzw. 0,25 W Verlustleistung. In der Regel ist das bei kleinen Spannungen bis 9 V und kleine Ströme bis 25 mA unproblematisch.

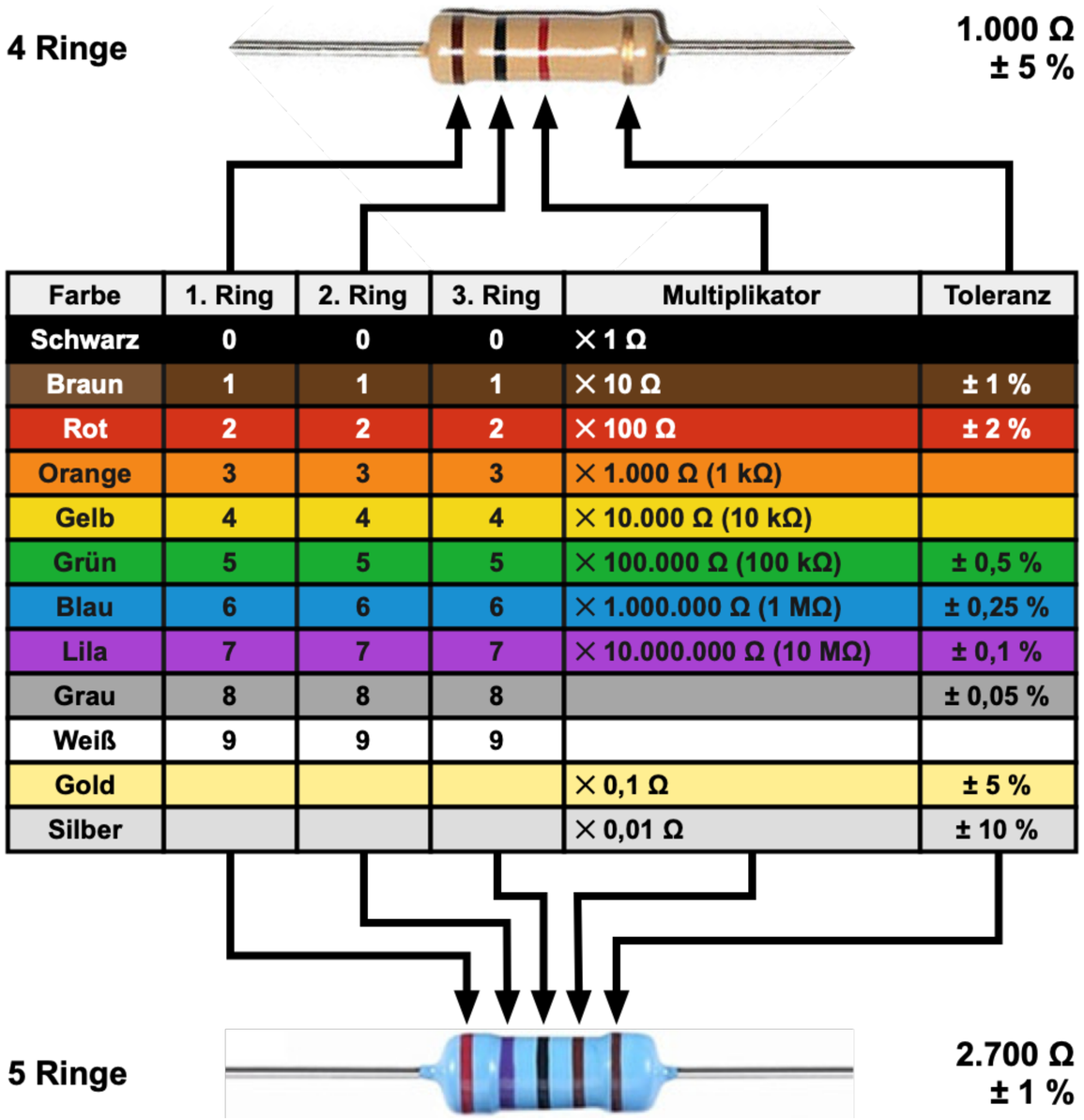
Hinweis: Widerstände, die mit einer zu großen Leistung betrieben werden, werden heiß und brennen durch. Dann qualmt es etwas und der Widerstand färbt sich braun bis schwarz.

Um einen bestimmten Widerstandswert zu errechnen, verwendet man das Ohmsche Gesetz.

In der praktischen Elektronik unterscheidet man zwischen Kohleschichtwiderständen und Metallfilmwiderständen.

Kohleschichtwiderstände haben meist einen hell gefärbten Widerstandskörper und weisen typischerweise 4 Ringe auf. Metallfilmwiderstände haben meist einen Hellblau gefärbten Widerstandskörper und weisen typischerweise 5 Ringe auf.

Kennzeichnung von Widerständen (1)



Farbe	1. Ring	2. Ring	3. Ring	Multiplikator	Toleranz
Schwarz	0	0	0	× 1 Ω	
Braun	1	1	1	× 10 Ω	± 1 %
Rot	2	2	2	× 100 Ω	± 2 %
Orange	3	3	3	× 1.000 Ω (1 kΩ)	
Gelb	4	4	4	× 10.000 Ω (10 kΩ)	
Grün	5	5	5	× 100.000 Ω (100 kΩ)	± 0,5 %
Blau	6	6	6	× 1.000.000 Ω (1 MΩ)	± 0,25 %
Lila	7	7	7	× 10.000.000 Ω (10 MΩ)	± 0,1 %
Grau	8	8	8		± 0,05 %
Weiß	9	9	9		
Gold				× 0,1 Ω	± 5 %
Silber				× 0,01 Ω	± 10 %

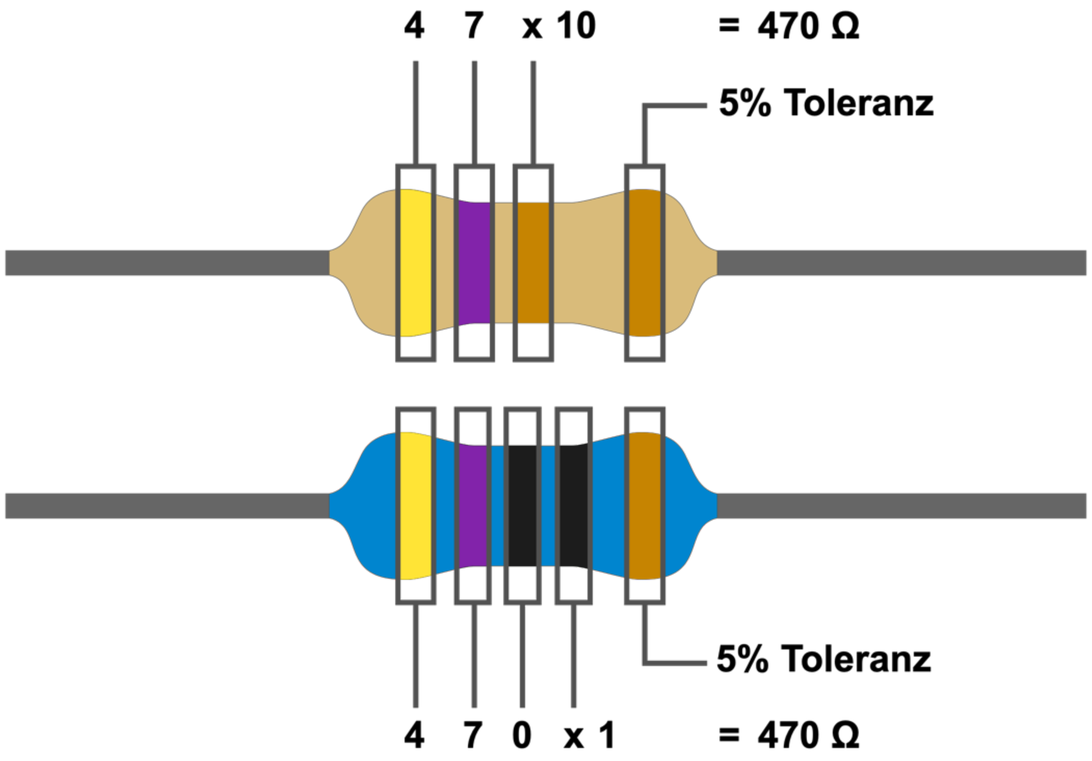
Widerstände werden in der Regel durch Farbringe gekennzeichnet. Je nach Art des Widerstandsmaterials, werden 4 oder 5 Ringe verwendet.

Kohleschichtwiderstände haben meist einen Beige (Mischung aus Weiß und Braun) gefärbten Widerstandskörper und weisen typischerweise 4 Ringe auf. Metallfilmwiderstände haben meist einen Hellblau gefärbten Widerstandskörper und weisen typischerweise 5 Ringe auf.

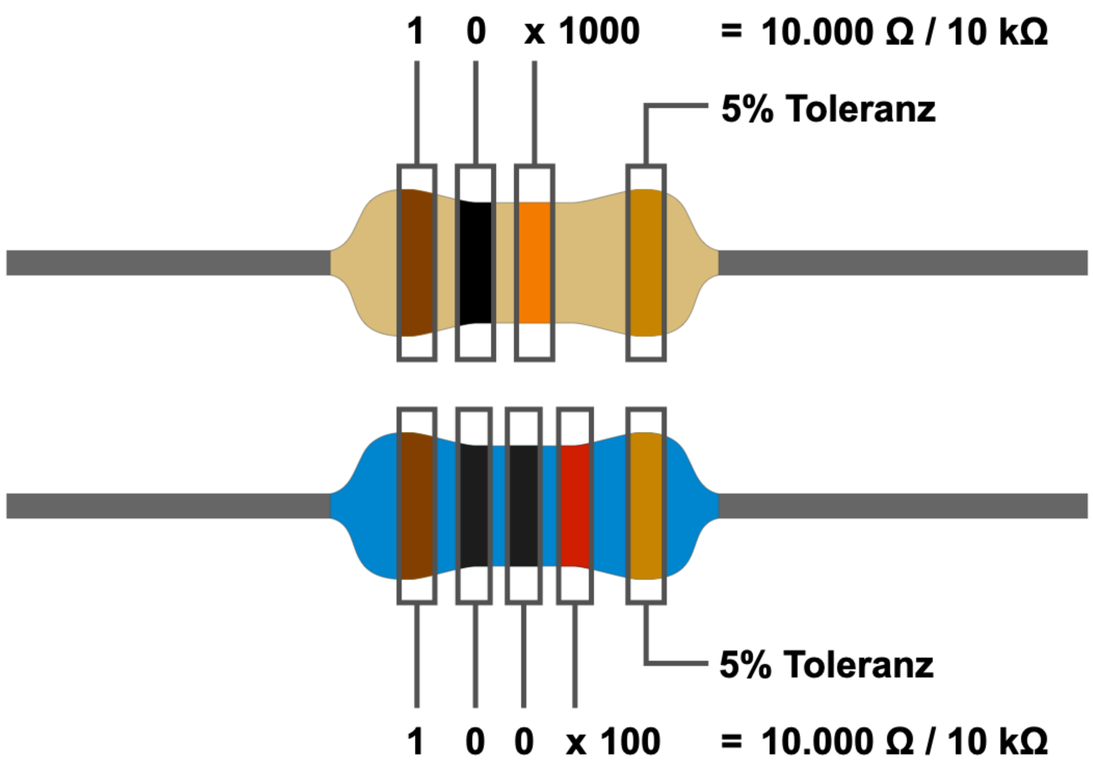
Die Ringe entsprechen einem internationalen Farbcode zur Kennzeichnung und Bestimmung des Widerstandswertes in Ω, sowie der Toleranz dieses Wertes. Um die Werte abzulesen, behilft man sich mit einer Tabelle.

Kennzeichnung von Widerständen (2)

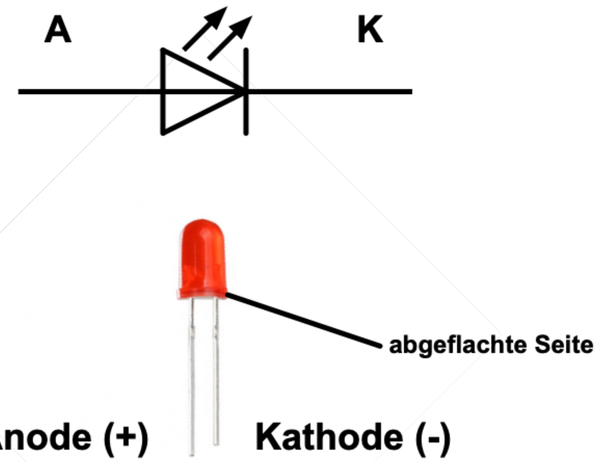
Widerstand mit 470 Ohm



Widerstand mit 10 kOhm



Leuchtdiode (LED)



- optischer Signalgeber
- optischer Sensor

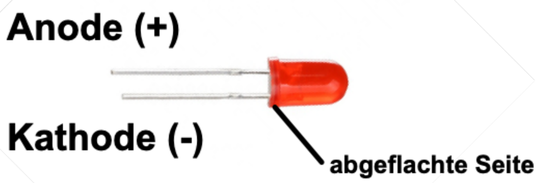
Eine Leuchtdiode, auch Light Emitting Diode, kurz LED genannt, erzeugt ein Licht in einer bestimmten Farbe, wenn sie von einem Strom durchflossen wird. Dabei verhält sich die LED wie jede andere Halbleiterdiode auch.

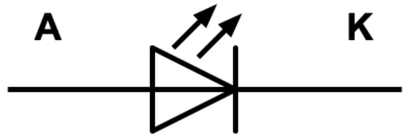
Die beiden Anschlüsse werden als Kathode und Anode bezeichnet. Typischerweise sind die beiden Anschlussdrähte einer LED unterschiedlich lang. Der längere von beiden ist die Anode. Der kürzere die Kathode. Außerdem sind die meisten LEDs auf der Kathodenseite abgeflacht. Um das zu erkennen, musst Du ganz genau hinschauen.

Leuchtdioden werden typischerweise mit einem Vorwiderstand in Reihe betrieben, der eine spannungs- und strombegrenzende Wirkung hat. Die Strombegrenzung ist deshalb notwendig, weil der LED-Halbleiter bei zu viel Strom kaputt gehen kann.

Leuchtdioden gibt es in vielen verschiedenen Farben. Am häufigsten kommen Rot, Grün, Gelb, Blau und Weiß vor. Die Farbe wird durch das Halbleitermaterial vorgegeben und zusätzlich ein entsprechend gefärbtes und lichtdurchlässiges Gehäuse verwendet.

Kennzeichnung und Anschlussbelegung von Leuchtdioden





Je nach Hersteller, Farbe und Typ können Leuchtdioden sehr unterschiedliche elektrische Werte aufweisen. Gemeint sind die Durchlassspannung und der Durchlassstrom. Beide Werte sind wichtig zur Berechnung des strombegrenzenden Vorwiderstands.

Ein Vorwiderstand begrenzt den Strom in Durchlassrichtung typischerweise auf etwa 10 mA. Die Spannung je nach LED-Typ auf 1,8 bis 2,2 Volt, oder höher. Manche LEDs haben eine Durchlassspannung von 3 oder sogar 4 Volt.

An 9 Volt empfiehlt sich ein Widerstandswert zwischen 1 kOhm bis 20 kOhm.

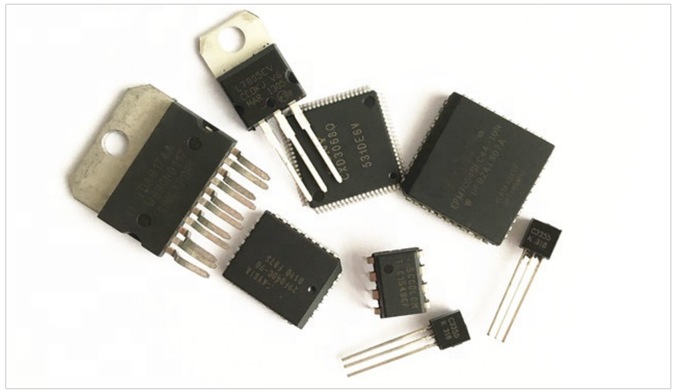
Bei einem deutlich kleineren Widerstand als 1 kOhm kann die LED kaputt gehen. Bei einem deutlich zu großen Widerstand als 20 kOhm leuchtet die LED zu schwach oder gar nicht.

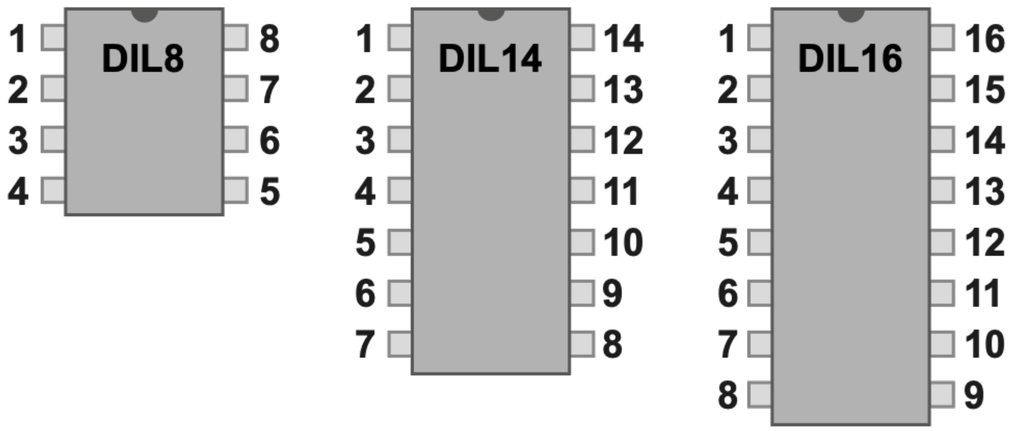
Weil sich eine LED wie jede andere Halbleiterdiode verhält, gibt es eine Sperrrichtung und eine Durchlassrichtung. Soll eine LED leuchten, muss sie in Durchlassrichtung angeschlossen sein. Also die Kathode (K) an Minus (-) und die Anode (A) an Plus (+).

Einfach zu merken: Das Pluszeichen hat einen Strich mehr als das Minuszeichen und deshalb ist der Anschlussdraht der Anode etwas länger. Außerdem sind die meisten LEDs auf der Minusseite abgeflacht, wie eben ein Minus, oder das "K" der Kathode.

Beim Schaltzeichen kann man sich das so merken: Das Schaltzeichen hat wegen dem Querbalken die Form des Buchstabens "K". Das Dreieck hat eine Ähnlichkeit mit dem Buchstaben "A". Beim Querbalken ist der Anschluss die Kathode und auf der anderen Seite die Anode. Die Anode zeigt vom Pluspol weg und zum Minuspol hin, was der technischen Stromrichtung entspricht. Und somit wird die Anode am Pluspol und die Kathode am Minuspol angeschlossen.

Integrierte Schaltkreise / Integrated Circuits (IC)





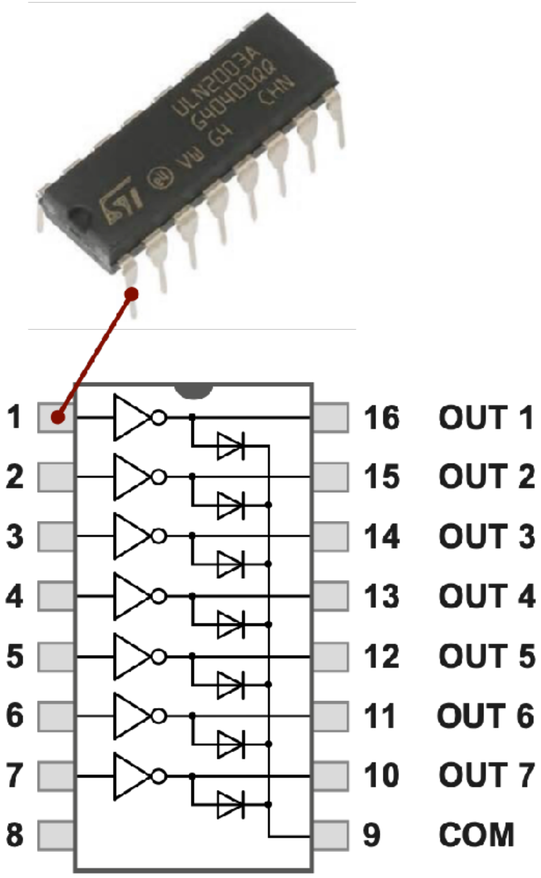
Wenn man auf die beschriftete Seite des ICs schaut, kann man an einer kurzen Seite eine Einkerbung entdecken. Diese Einkerbung kennzeichnet „oben“. Links von der Einkerbung beginnt die Nummerierung der Pins von 1 an nach unten gegen den Uhrzeigersinn zu zählen.

- komplexe Standard-Schaltung als ein Bauteil
- Platz und Kosten sparen durch mehrfach integrierte Funktionen
- geringerer Schaltungsaufwand

Viele Schaltungen oder Schaltungsteile kommen in der praktischen Elektronik immer wieder vor. Um diese teilweise komplexen Schaltungen nicht immer wieder neu aufbauen oder erfinden zu müssen, werden sie in eine integrierte Schaltung (IS = Integrierter Schaltkreis) zusammengefasst und in einem Gehäuse vergossen.

Fast alle ICs gibt es in unterschiedlichen Gehäuseformen. Die unterscheiden sich nicht nur in der Größe, sondern hauptsächlich in der Art der Anschlusspins und wie die mit der restlichen Schaltung verbunden werden. Also beispielsweise, ob das IC löt- oder steckbar ist. In der Hobby-Elektronik werden ICs gerne mit DIL- bzw. DIP-Gehäuse verwendet. Diese eignen sich zum Stecken in eine Fassung und zum Verlöten auf eine Platine.

ULN2003A - Darlington Transistor Array (IC)



- Eingänge für TTL-Signale (5 Volt)
- Ausgänge für bis 50 Volt
- Ausgänge mit Freilaufdiode für induktive Lasten (z. B. Motor und Relais)

Ein ULN2003A ist ein integrierter Schaltkreis mit 7 bipolaren NPN-Darlington-Transistoren mit offenem Kollektor und gemeinsamen Emitter. Damit kann man eine Spannung bis 50 Volt (V) und einen Strom bis 500 Milliampere (mA) pro Ausgang schalten. Eine Besonderheit ist die bereits integrierte Freilaufdiode an den Ausgängen. Dadurch kann man problemlos Relais, Motoren und andere induktive Lasten schalten.

Die Eingänge eines ULN2003A eignen sich für die Ansteuerung mit TTL- (Transistor-Transistor-Logik) und CMOS-Signalen (Komplementär-Metalloxid-Halbleiter) bis maximal 5 Volt. mit einem Transistor beschaltet werden soll.

Wenn Du an einem Ausgang des ULN2003A ein Relais oder einen Motor anschließt, dann verbinde auch den COM-Anschluss (Pin 9) mit der Versorgungsspannung der angeschlossenen Verbraucher. Dadurch aktivierst Du die integrierten Freilaufdioden, die die Darlington-Stufen vor der Rückwirkung induktiver Lasten schützen.

ULN2003A: Beschaltung

Eingänge beschalten

Die Eingänge arbeiten mit Standard-TTL-Signalen. Also bei High-Pegel mit 5 Volt und bei Low-Pegel mit 0 Volt. Typischerweise wird ein TTL-Eingang eine anliegende Spannung bis 2 Volt runter als High-Pegel erkennen. Das bedeutet, dass ein High-Pegel eines GPIO-Ausgangs mit 3,3 Volt vom ULN2003A-Eingang als High-Pegel erkannt wird.

Ausgänge beschalten

Welche Spannung geschaltet werden soll hängt von den angeschlossenen Verbrauchern ab. Diese Spannung muss in der Regel aus einer eigenen Spannungsversorgung bereitgestellt werden.

Aufgrund der integrierten Darlington-Stufe haben die Ausgänge bei einen Low-Pegel nie ganz 0 Volt, sondern etwa 0,9 Volt. Unter Umständen ist das beim Dimensionieren nachfolgender Schaltungsteile zu berücksichtigen.

Werden alle 7 Transistorstufen dauerhaft durchgeschaltet (Tastverhältnis 100%), darf jede Stufe nur mit max. 160 mA belastet werden.

COM-Anschluss (Pin 9): Freilaufdiode aktivieren

Der Anschluss Pin 9 (COM) ist der gemeinsame Anschluss der integrierten Freilaufdioden. Dieser Anschluss muss nur dann beschaltet werden, wenn an den Ausgängen induktive Lasten angeschlossen sind. Zum Beispiel Relais oder Motoren. Sollte das der Fall sein, dann wird Pin 9 typischerweise mit der Versorgungsspannung der angeschlossenen Verbraucher verbunden.

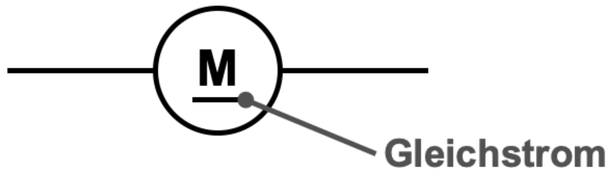
Masse bzw. Ground (GND)

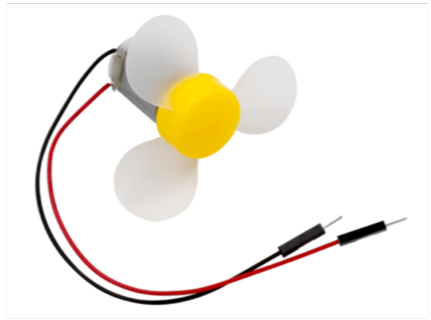
Der Anschluss Pin 8 ist der gemeinsame Masse-Anschluss bzw. Ground (GND) für die sieben Eingänge und die sieben Schaltstufen. In der Regel muss auch der Ground einer separaten Versorgungsspannung damit verbunden werden.

Hinweis

Es empfiehlt sich einen Blick ins Datenblatt zu werfen.

Motor für Gleichstrom (DC)





- Umwandlung von elektrischer Energie in Bewegungsenergie oder mechanische Energie
- typischerweise Bewegen von Fahrzeugen und Antrieben

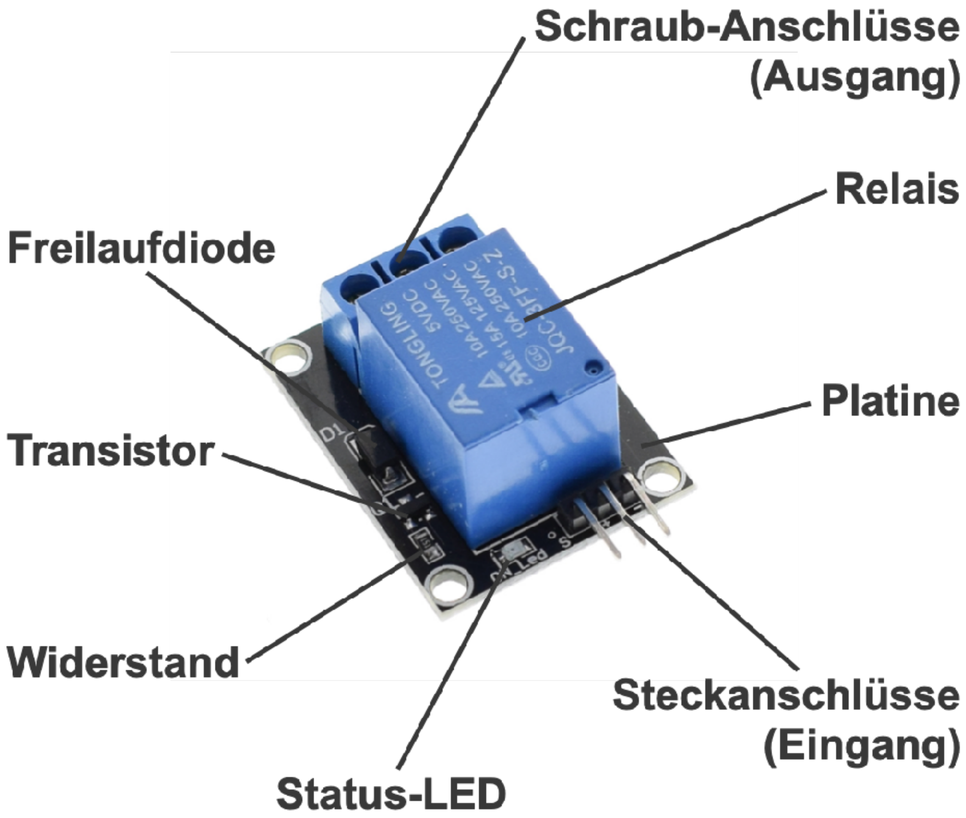
Gleichstrom-Motoren sind typischerweise gepolt. Deshalb sind die Anschlüsse mit unterschiedlichen Kabel- oder Adernfarben versehen. Typischerweise stellt die Farbe Rot den Plus-Pol dar. Die Farben Blau oder Schwarz den Minus-Pol. Bei einem Gleichstrom-Motor dienen diese Farben aber nur zur Unterscheidung der beiden Anschlüsse. Denn wie herum gepolt wird, hängt von der gewünschten Bewegungsrichtung ab. Eine Falschpolung wird nur funktional dazu führen, dass der Motor in die falsche Richtung dreht. Gemeint ist, er läuft je nach Polung vorwärts oder rückwärts. Je nach Sichtweise auch rechtsherum oder linksherum.
Du kannst einen Gleichstrom-Motor also durch Falschpolung nicht kaputt machen.

Motoren findet man überall dort, wo etwas bewegt werden muss. Zum Beispiel Fahrzeuge, Ventilatoren und Antriebe zum Öffnen und Schließen. In der Elektrotechnik und Elektronik werden entsprechend Elektro-Motoren verwendet. Das heißt, sie werden mit elektrischem Strom betrieben. Je nach Spannungsart, Spannungsbereich und Leistungsbedarf gibt es sehr viele unterschiedliche Typen.

Im einfachsten Fall handelt es sich um einen Gleichstrom-Motor (DC), der im Spannungsbereich von 3 bis 6 Volt mit etwa 180 Milliampere (mA) arbeitet.

Motoren und Antriebe laufen nicht dauerhaft, sondern nur wenn etwas bewegt werden soll. Zum manuellen Bedienen, also zum Einschalten und Ausschalten, eignen sich meist Taster oder Schalter in Reihe zum Motor.

Relais-Board (KY-019)



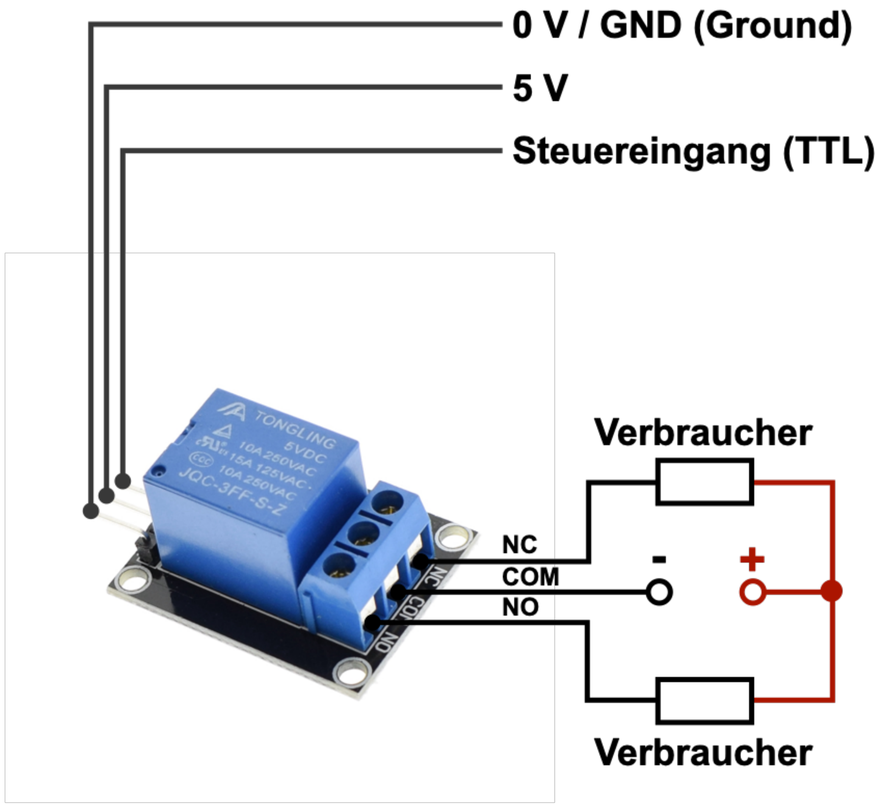
- potentialfreies Schalten
- integrierte Freilaufdiode
- schraubbare Anschlüsse für Leitungen zum schaltenden Gerät

Ein Relais findet oft dort Anwendung, wo eine galvanische Trennung zwischen Steuer- und Laststromkreis erforderlich ist. Oft kommt es vor, dass beide Stromkreis bezüglich Spannung und Strom nicht kompatibel sind. Man braucht also ein Bauteil oder einen Schaltkreis, der beide elektrisch voneinander trennt, aber funktional trotzdem eine Verbindung ermöglicht. Ein solches Bauteil kann ein Relais sein.

Um ein Relais anzusteuern braucht es nicht viel: Ein Transistor, zwei Widerstände und eine Diode. Welche genau das sind, hängt vom Relais ab. Der Vorteil eines Relais-Boards ist, dass die notwendigen Bauteile korrekt dimensioniert und auf der Platine mit dem Relais richtig verschaltet sind. Außerdem hat ein Relais-Board auch noch schraubbare Anschlüsse zum Verbinden mit der zu steuernden Schaltungen.

Auch wenn das Relais-Modul einfach anzuschließen ist, sind gerade deshalb ein paar Dinge zu beachten. Auch wenn die Beschriftung des Relais-Boards die Eignung bis 250 Volt ausweist, ist es nicht zwangsläufig dafür geeignet. Arbeiten mit 230 Volt sind gefährlich und können tödlich sein. Die Eignung und der sichere Betrieb des Moduls bei 230 Volt können nur Fachleute gewährleisten.

Relais-Board beschalten (1)

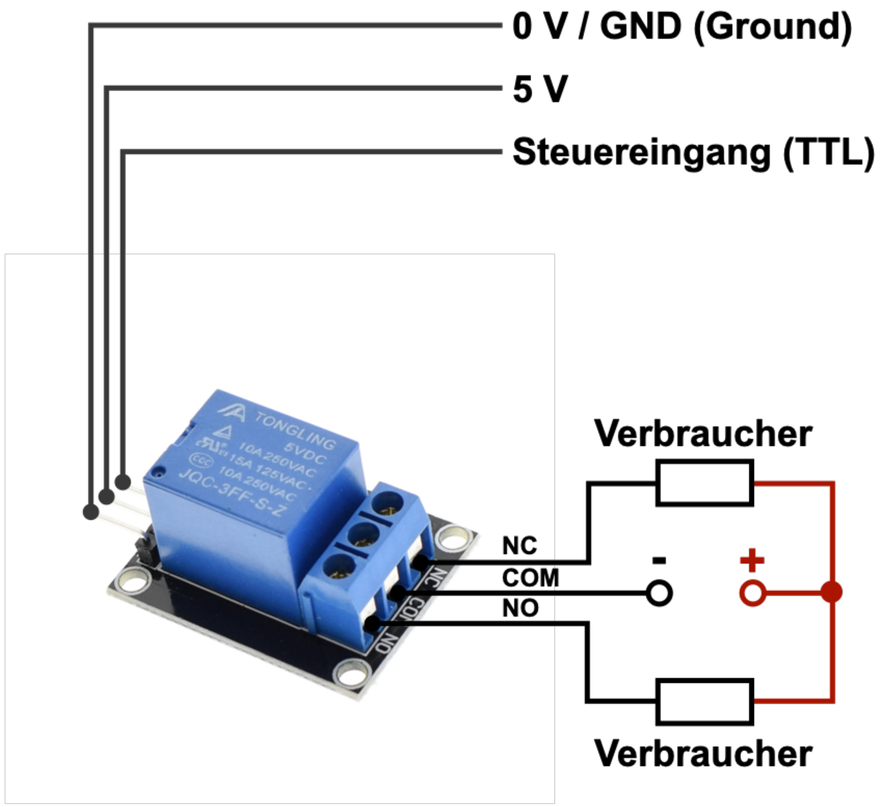


Beschaltung der Eingangsseite vom GPIO

Die Eingangsseite des Relais-Boards hat 3 Anschlüsse in Form von Pins bzw. einer Pin-Leiste. An zwei Pins wird typischerweise die Betriebsspannung angeschlossen. Die ist abhängig vom Relais. Die meisten Relais-Boards sind TTL-kompatibel und haben eine Betriebsspannung von 5 Volt. Deshalb wird am Pin, der mit einem Plus (+) oder VCC gekennzeichnet ist, die 5V-Leitung angeschlossen und am Pin, der mit Minus (-), 0V oder GND gekennzeichnet ist entsprechend mit der 0V-Leitung bzw. GND verbunden.

Ein Relais-Board verfügt zusätzlich über einen Steuereingang, mit dem das Relais geschaltet wird. Der Steuereingang ist typischerweise mit „S“, „IN“ oder ähnlichen Zeichen gekennzeichnet, die von der Kennzeichnung der Betriebsspannung abweichen.

Relais-Board beschalten (2)

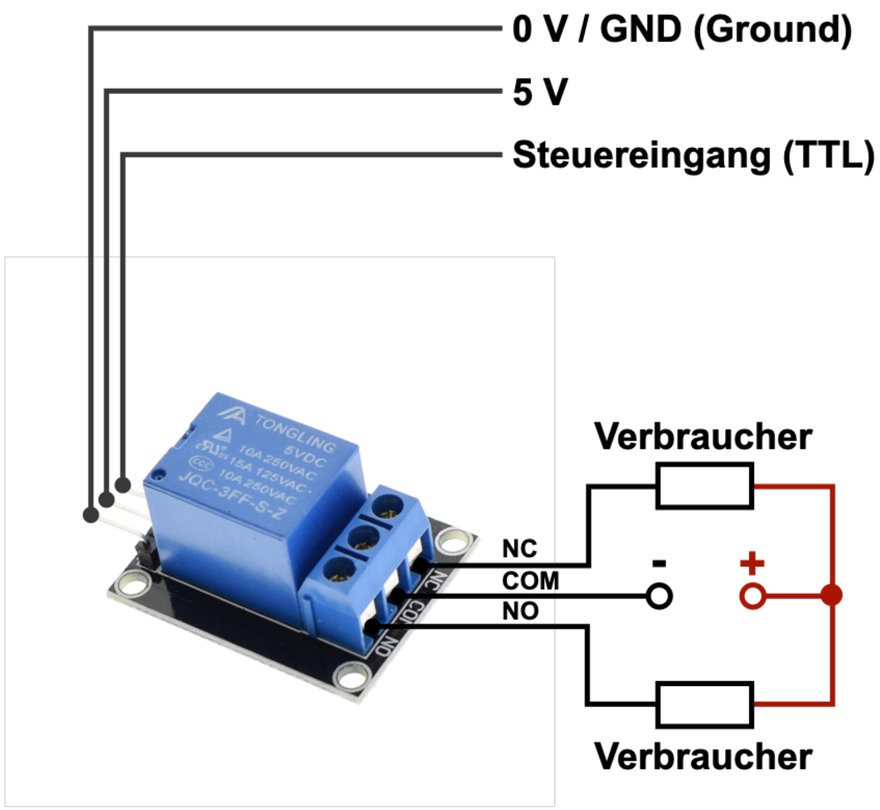


Beschaltung der Ausgangsseite zum schaltenden Gerät

Die Beschaltung der Ausgangsseite erfolgt meist über schraubbare Anschlussklemmen. Dahinter verbergen sich die Arbeitskontakte, die je nach Relais unterschiedliche Funktionen aufweisen können. Die typischen Relais-Boards haben nur zwei Funktionen, die entsprechend gekennzeichnet sind:

- Schließer (Normally Open, NO): Der Schließer ist im Ruhezustand geöffnet. Erst wenn im Steuerstromkreis ein Strom fließt, dann schließt sich dieser Arbeitskontakt.
- Öffner (Normally Closed, NC): Der Öffner ist im Ruhezustand geschlossen. Erst wenn im Steuerstromkreis ein Strom fließt, dann öffnen sich dieser Arbeitskontakt.
- Beide Arbeitskontakte haben einen gemeinsamen Anschlusspunkt (COM), an dem die andere Seite des Stromkreises angeschlossen werden muss.

Relais-Board beschalten (3)



Wie sind die drei Anschlüsse funktional miteinander verbunden?

- Im spannungslosen Zustand ist der Schaltkontakt zwischen COM und NC geschlossen und der Schaltkontakt zwischen COM und NO offen.
- Im Fall einer anliegenden Spannung ist der Schaltkontakt zwischen COM und NC geöffnet und der Schaltkontakt zwischen COM und NO geschlossen.

Wie sollte ein Gerät oder eine Schaltung angeschlossen werden?

Wenn ein Gerät im Normal- bzw. Grundzustand ausgeschaltet ist, dann wäre das Gerät in Reihe zu COM und NO anzuschließen. Wäre das Gerät im Normalzustand eingeschaltet, dann wäre das Gerät in Reihe zu COM und NC anzuschließen.

Experimente und Anwendungen

Experimente und Anwendungen mit Python und GPIO Zero

- LED einschalten und ausschalten
- LED blinken lassen
- LED-Helligkeit steuern
- Taster-Zustand anzeigen
- LED mit Taster einschalten und ausschalten
- LED über ULN2003A steuern
- Motor über ULN2003A steuern
- Relais-Board über ULN2003A steuern
- Relais-Board ohne ULN2003A Steuern
- LED-Ampel-Steuerung
- LED-Lauflicht
- Raspberry Pi per Taster herunterfahren
- Einfaches Reaktionsspiel für zwei Personen

Python ist eine universelle Programmiersprache. Sie gilt als einfach zu erlernende Sprache, da sie über eine klare und übersichtliche Syntax und eine gute Lesbarkeit verfügt. Des Weiteren ist Python in der Informatik weit verbreitet, so dass man sie in Technik-nahen Ausbildungen und Berufen immer wieder findet.

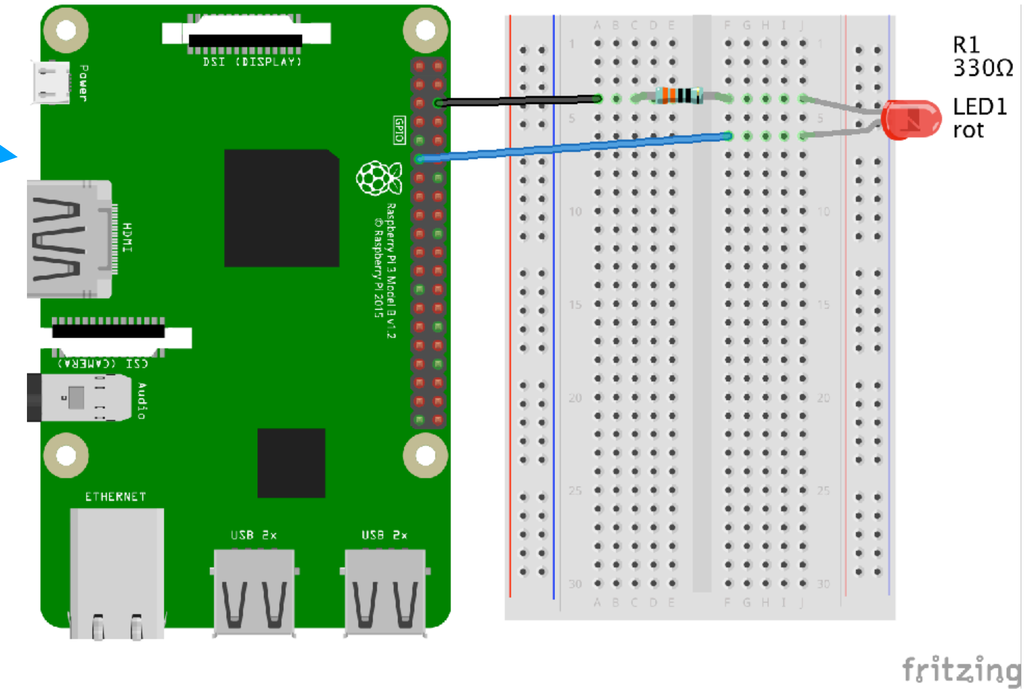
GPIO Zero ist eine Python-Bibliothek. Sie stellt eine Software-Schnittstelle für Physical Computing bereit, um LEDs, Taster, LED-Reihen, 7-Segment-Anzeigen, Analog-Digital-Wandler und vieles mehr über die GPIOs in Python zu programmieren und zu steuern.

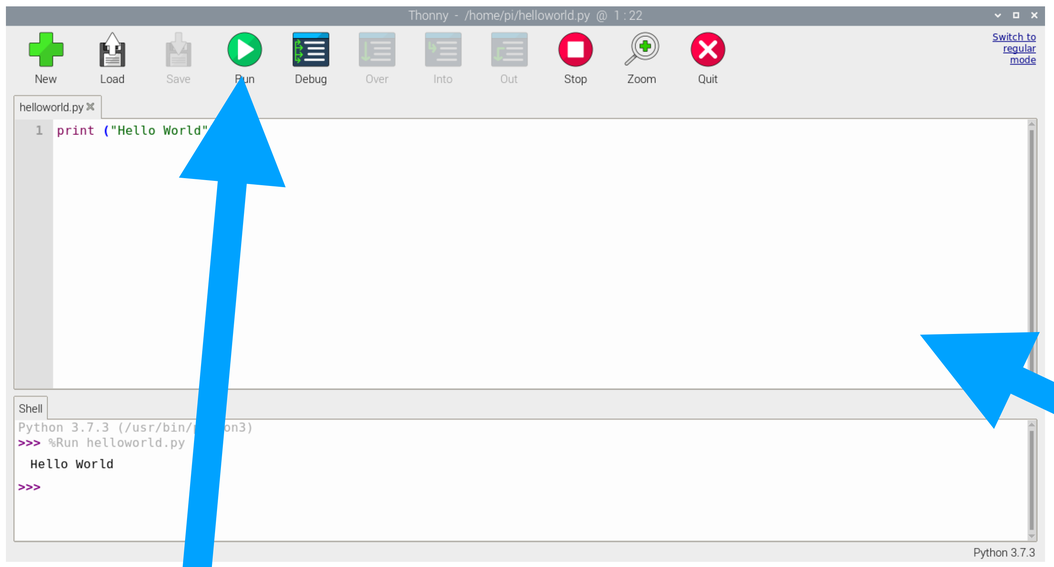
Die folgenden Experimente bauen auf der offizielle Dokumentation von GPIO Zero auf. Auf den folgenden Seiten findest Du einige Beispiele aus der Dokumentation mit ein paar zusätzlichen Erklärungen und wie sie sinnvoll einzusetzen sind.

Bei den folgenden Experimenten handelt es sich um keine sinnvollen und vollständig laufenden Anwendungen, sondern nur um Beispiele, wie die GPIOs für angeschlossene Bauteile konfiguriert und deren Zustände gesetzt und gelesen werden können.

Vorgehensweise

1. **Zuerst die Schaltung aufbauen**





2. **Dann den Programmcode ins Textfeld kopieren und speichern**

```
from gpiozero import LED
from time import sleep

red = LED(17)

red.on()
sleep(5)
red.off()
```

3. **Danach das Programm ausführen**

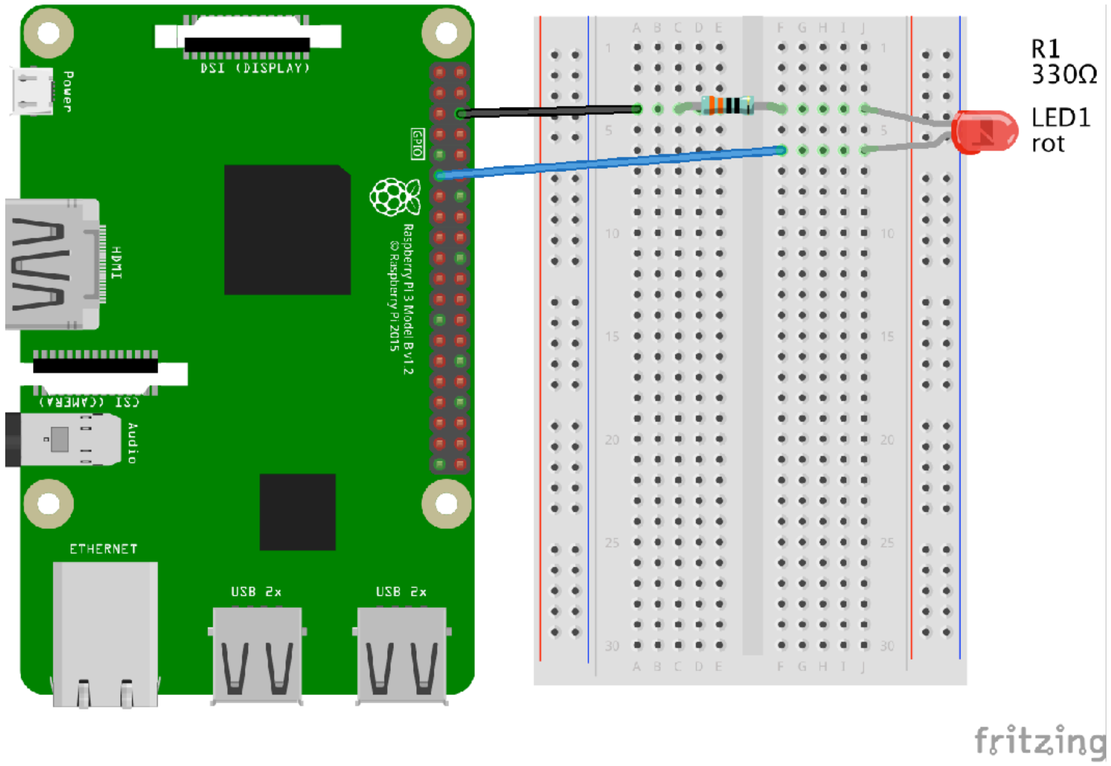
4. **Funktion von Aufbau und Programm prüfen**

LED einschalten und ausschalten

In diesem Aufbau geht es darum, an einem Raspberry Pi eine LED richtig anzuschließen und mit einem Programm die LED einzuschalten und wieder auszuschalten. Das ist nicht weiter schwer. Wir brauchen dazu eine LED (egal welche Farbe) und einen Widerstand. Auf einem Steckbrett werden die Raspberry-Pi-GPIO-Pins und die Bauteile mit den Verbindungskabeln verbunden.

Die folgende Schaltung und das Programm sind denkbar einfach. Es ermöglicht Dir, Dich mit dem Verbinden von nur zwei Bauteilen auf dem Steckbrett vertraut zu machen und ein erstes Programm mit Python und GPIO Zero zu schreiben.

Wenn Du das alles geschafft hast, dann können Dich die nächsten Experimente nicht mehr schrecken.
Es bietet sich an, anders farbige Leuchtdioden auszuprobieren.

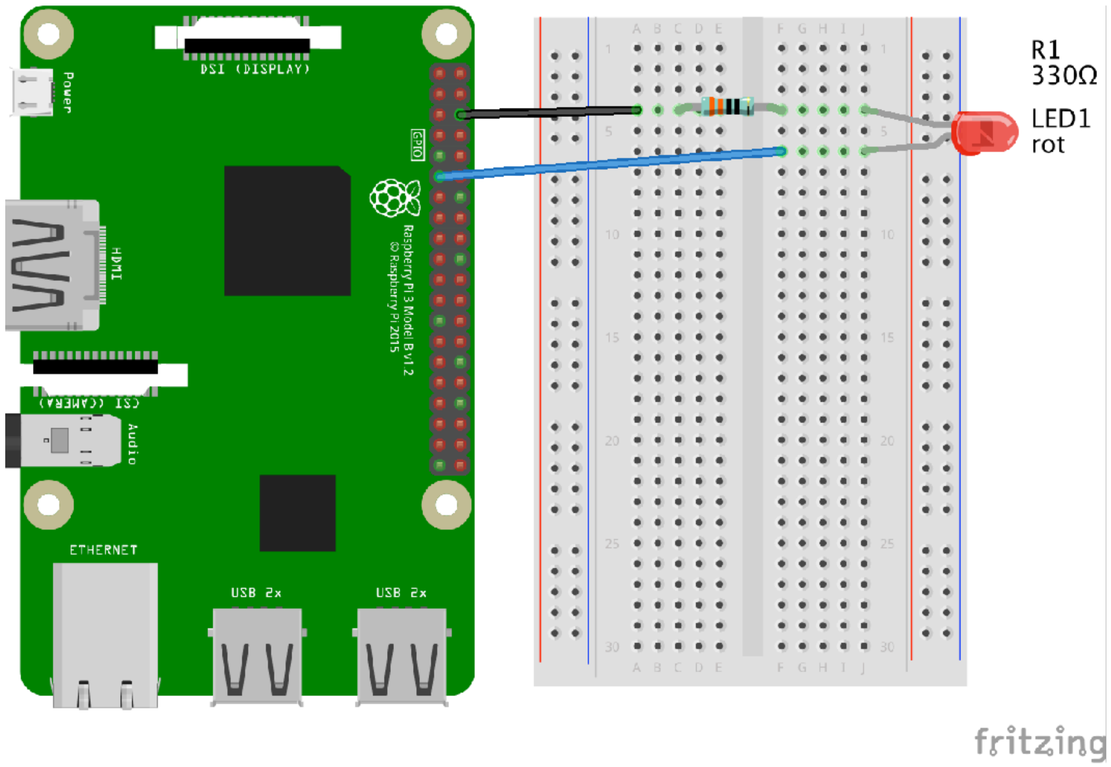



```
from gpiozero import LED
from time import sleep

# Initialisierung von GPIO17 als LED (Ausgang)
led = LED(17)

# LED: ein / 5 Sekunden warten / aus
led.on()
sleep(5)
led.off()
```

LED blinken lassen (1)

Am Besten ist Elektronik immer dann, wenn es blinkt und blitzt. Aber bitte nicht zu viel davon. Wenn es zu sehr blitzt, ist danach meistens etwas kaputt. Deshalb beschränken wir uns sicherheitshalber auf das Blinken einer Leuchtdiode.

Mit Python und GPIO Zero lässt sich das Blinken einer Leuchtdiode mit wenigen Zeilen erledigen.



Damit eine LED blinkt oder blitzt gibt es in der Elektronik nur ein paar Grundschaltungen, die man je nach Blinkfrequenz variieren kann oder sogar bevorzugt. Die folgenden Programmcodes ersetzen praktisch die Funktion dieser Grundschaltungen. Das einzige, was an Hardware bleibt, ist die Leuchtdiode mit Vorwiderstand. Und natürlich der Raspberry Pi.

Die Leuchtdiode wird per Software gesteuert. Dabei gibt es nicht nur einen Weg, wie man die Leuchtdiode zum Blinken bringt. Es gibt gleich mehrere davon und auch noch in verschiedenen Variationen. Und genau das sollst Du ausprobieren.

In der ersten Variante erfolgt das Blinken der Leuchtdiode durch hartes Ein- und Ausschalten.

```
from gpiozero import LED

# Initialisierung von GPIO17 als LED (Ausgang)
led = LED(17)

# LED blinken lassen
led.blink()
```

LED blinken lassen (2)

In der zweiten Variante erfolgt das Blinken durch weiches Ein- und Ausblenden, in dem die Helligkeit über ein PWM-Signal gesteuert wird.

```
from gpiozero import PWMLED

# Initialisierung von GPIO17 als PWM-Signal (LED)
led = PWMLED(17)

# PWM-Signal ausgeben
led.pulse()
```

In der dritten Variante wird das Blinken mit unterschiedlicher Leuchtdauer und Pausedauer realisiert. Die Werte von „sleep“ sind die Dauer in Sekunden.

```
from gpiozero import LED
from time import sleep

# Initialisierung von GPIO17 als LED (Ausgang)
led = LED(17)

# Wiederholung einleiten
while True:
    # LED einschalten
    led.on()
    # 0,17 Sekunden warten
    sleep(0.17)
    # LED ausschalten
    led.off()
    # 1 Sekunde warten
    sleep(1)
```

LED blinken lassen (3)

In der vierten Variante wird das Blinken mit unterschiedlicher Leuchtdauer und Pausedauer realisiert. Die Werte von „blink“ sind die Dauer in Sekunden.

```
from gpiozero import LED

# Initialisierung von GPIO17 als LED (Ausgang)
led = LED(17)

# LED im Verhältnis 1 Sekunde zu 2 Sekunden
blinken lassen
led.blink(1,2)
```

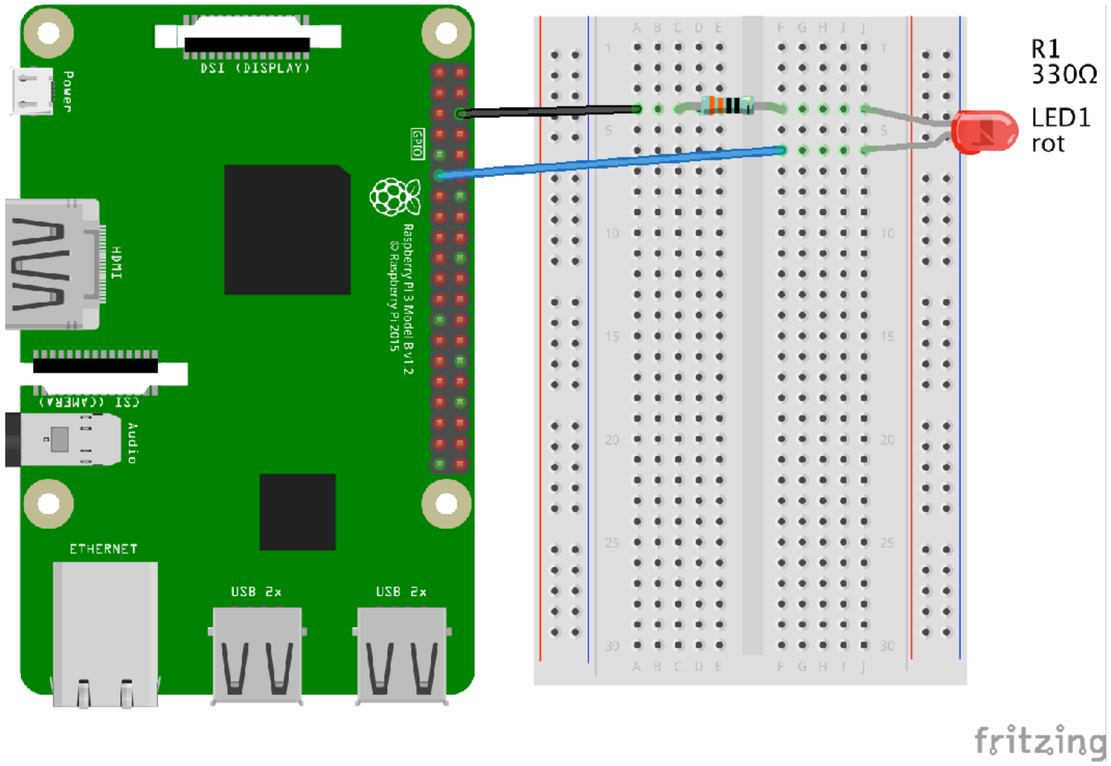
Die letzten beiden Beispiele erlauben verschiedene Experimente:

1. Versuche verschiedene Kombination mit unterschiedlichen Leucht- und Pausezeiten aus. Zum Beispiel 0.5 und 2.
2. Welche kleinste Zeit kann man als Leuchtdauer angeben, damit man die Leuchtdiode noch leuchten sieht?
3. Welche kleinste Zeit kann man als Pausedauer angeben, damit die Leuchtdiode noch ausgeht?

LED-Helligkeit steuern

Im Prinzip kennt ein GPIO nur zwei Zustände: „High“ und „Low“. Also „An“ und „Aus“ oder „1“ und „0“. Es handelt sich dabei um die binäre Logik. Werte dazwischen gibt es nicht. Logisch wäre es, dass man mit der Steuerung per GPIO eine LED auch nur ein- und ausschalten kann. Die Helligkeit kann man also nicht einstellen.

Trotzdem gibt es einen Trick mit dem man die Helligkeit einer LED doch steuern kann. Wenn die LED sehr schnell ein- und ausgeschaltet wird, also ein Taktsignal ausgeben wird. Dieses Taktsignal wird als PWM-Signal realisiert, bei dem die Breite des Pulses über die Zeit eingestellt werden kann. Das heißt, es wird die Dauer des Leuchtens der LED eingestellt. Durch die Trägheit des Aufleuchtens und der Wahrnehmung des menschlichen Auges sieht es so aus, als ob die Helligkeit der Leuchtdiode gesteuert wird.



LED-Helligkeit steuern (2)

Mit Python und GPIO Zero lässt sich ein PWM-Signal mit wenigen Zeilen erzeugen. Auf diese Weise kann man die Helligkeit einer Leuchtdiode einstellen.

Nach dem Start des Programms wird die Leuchtdiode in 10 Stufen (von 0 bis 1) von „aus“ nach „ganz hell“ geändert. Das Programm endet im Prinzip nie. Es muss manuell beendet bzw. gestoppt werden.

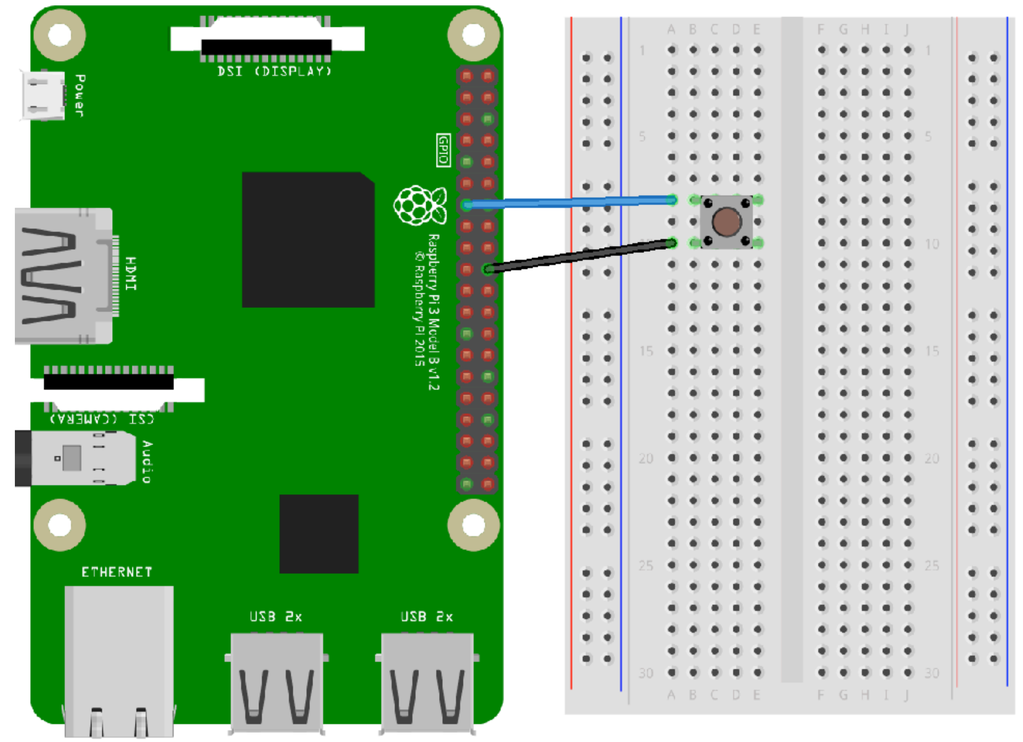
```
from gpiozero import PWMLED
from time import sleep

# Initialisierung von GPIO17 als PWM-Signal (LED)
led = PWMLED(17)

# Wiederholung einleiten
while True:
    # LED ausschalten
    led.value = 0
    sleep(1)
    led.value = 0.1
    sleep(1)
    led.value = 0.25
    sleep(1)
    # LED mit halber Helligkeit
    led.value = 0.5
    sleep(1)
    led.value = 0.75
    sleep(1)
    led.value = 0.9
    sleep(1)
    # LED mit voller Helligkeit
    led.value = 1
    sleep(1)
```

Taster-Zustand auswerten und anzeigen (1)

Ein Taster kann zwei Zustände haben: „gedrückt“ und „nicht gedrückt“. Entsprechend für „Ein“ und „Aus“. Die folgenden Programmbeispiele berücksichtigen verschiedene Szenarien in denen ein Taster gedrückt oder auch wieder losgelassen wird.

Zu allem Überfluss macht es einen Unterschied, ob man einen Taster nach Ground (GND) oder nach +3,3V (VCC) schaltet. Beim Experimentieren ist es eigentlich egal und macht nur im Programmcode beim Initialisieren des GPIOs einen Unterschied. In der praktischen Anwendung muss man dann sehen, was sinnvoller ist, um einen stabilen und fehlerfreien Betrieb zu gewährleisten.

GPIO Zero kennt mehrere Möglichkeiten die Betätigung eines Tasters auszuwerten, der nach Ground (GND) geschaltet wird.
Das folgende Programm wird angehalten und läuft erst dann weiter, wenn der Taster gedrückt wird. Nachdem der Text ausgegeben wurde, wird das Programm beendet.




```
from gpiozero import Button

# Initialisierung von GPIO27 als Button (Eingang)
button = Button(27)

# Warten auf Druck auf Button
button.wait_for_press()

# Text-Ausgabe
print("Taster wurde gedrückt")
```

Taster-Zustand auswerten und anzeigen (2)

Im folgenden Programm wird der Druck auf den Taster immer wieder erneut geprüft. Der Nachteil davon ist, dass die Betätigung mehrmals erkannt wird, auch wenn man nur einmal drückt. Das macht natürlich nur Sinn, wenn man wissen will, wenn der Taster nicht nur gedrückt wurde, sondern auch gehalten wird.

```
from gpiozero import Button

# Initialisierung von GPIO27 als Button (Eingang)
button = Button(27)

# Wiederholung einleiten
while True:
    # Wenn Button gedrückt wurde
    if button.is_pressed:
        # Text-Ausgabe
        print("Taster wurde gedrückt")
```

Im folgenden Programm wird die Auswertung des Tasters im Programmablauf global definiert. Das heißt, immer dann, wenn der Taster gedrückt wird, wird der Text ausgegeben. Das Programm wird nicht automatisch beendet.

```
from gpiozero import Button

# Definition einer Funktion
def pressed():
    # Text-Ausgabe
    print("Taster wurde gedrückt")

# Initialisierung von GPIO27 als Button (Eingang)
button = Button(27)

# Wenn der Button gedrückt wird
button.when_pressed = pressed
```

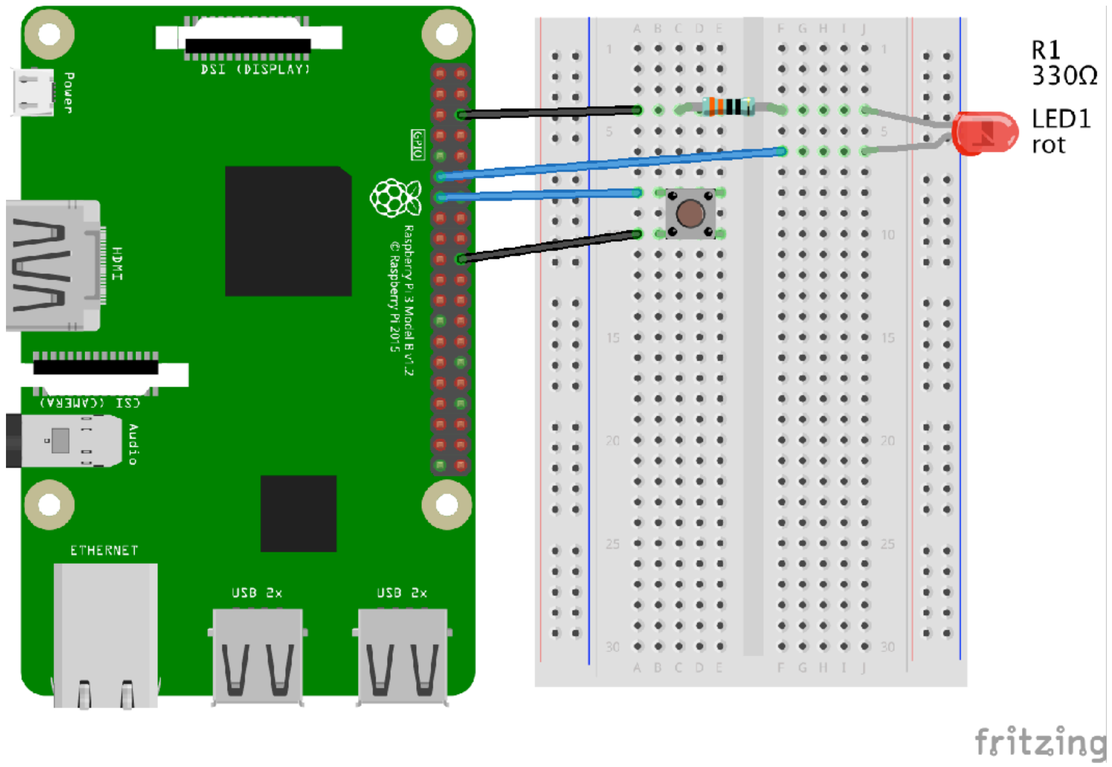
Taster-Zustand auswerten und anzeigen (3)

Im folgenden Programm wird die Auswertung des Tasters zusätzlich um das Loslassen im Programmablauf global definiert. Das heißt, immer dann, wenn der Taster gedrückt und losgelassen wird, wird ein Text ausgegeben. Das Programm wird nicht automatisch beendet.

```
from gpiozero import Button

# Definition einer Funktion
def pressed():
    # Text-Ausgabe
    print("Gedrückt")

# Definition einer Funktion
def released():
    # Textausgabe
    print("Losgelassen")

# Initialisierung von GPIO27 als Button (Eingang)
button = Button(27)

# Wenn der Button gedrückt wird
button.when_pressed = pressed

# Wenn der Button losgelassen wird
button.when_released = released
```

LED mit Taster einschalten und ausschalten (1)

Damit eine Leuchtdiode (LED) beim Drücken eines Tasters ein- und ausgeht braucht man eigentlich keine Softwaresteuerung. Diese Funktion kann man natürlich auch ohne Software realisieren. Einfach nur dadurch, dass der Taster mit der Leuchtdiode und einem Vorwiderstand in Reihe geschaltet wird.



Allerdings kann der Taster dann auch nur diese eine festgelegte Funktion. Soll der Taster eine andere Funktion haben, dann muss man die Hardware ändern. Wenn die Funktion fest verdrahtet oder gelötet ist, dann ist eine Änderung nicht so einfach möglich.

Es gibt also gute Gründe, warum heute alles mit Software realisiert wird und prozessorgesteuert ist. Eine Funktion lässt sich in der Software einfach leichter ändern als in der Hardware.

Dazu erweitern wir das Programm in einem zweiten Versuch um das Speichern des Zustandes (Haltefunktion). Das heißt, drücken wir den Taster wird die LED eingeschaltet. Drücken wir ein zweites Mal wird die LED ausgeschaltet. Man spricht hier auch von einer

Taster-Zustand auswerten und anzeigen (2)

Das folgende Programm schaltet die LED immer dann ein, wenn der Taster gedrückt wird und aus, wenn der Taster losgelassen wird. Das Programm wird nicht automatisch beendet.

```
from gpiozero import Button, LED

# Initialisierung von GPIO27 als Button (Eingang)
button = Button(27)

# Initialisierung von GPIO17 als LED (Ausgang)
led = LED(17)

# Wenn der Button gedrückt wird
button.when_pressed = led.on

# Wenn der Button losgelassen wird
button.when_released = led.off
```

Das folgende Programm schaltet die LED immer dann ein und wieder aus, wenn der Taster gedrückt wird. Das Programm wird nicht automatisch beendet.

```
from gpiozero import Button, LED

# Initialisierung von GPIO27 als Button (Eingang)
button = Button(27)

# Initialisierung von GPIO17 als LED (Ausgang)
led = LED(17)

# Wenn der Button gedrückt wird
button.when_pressed = led.toggle
```

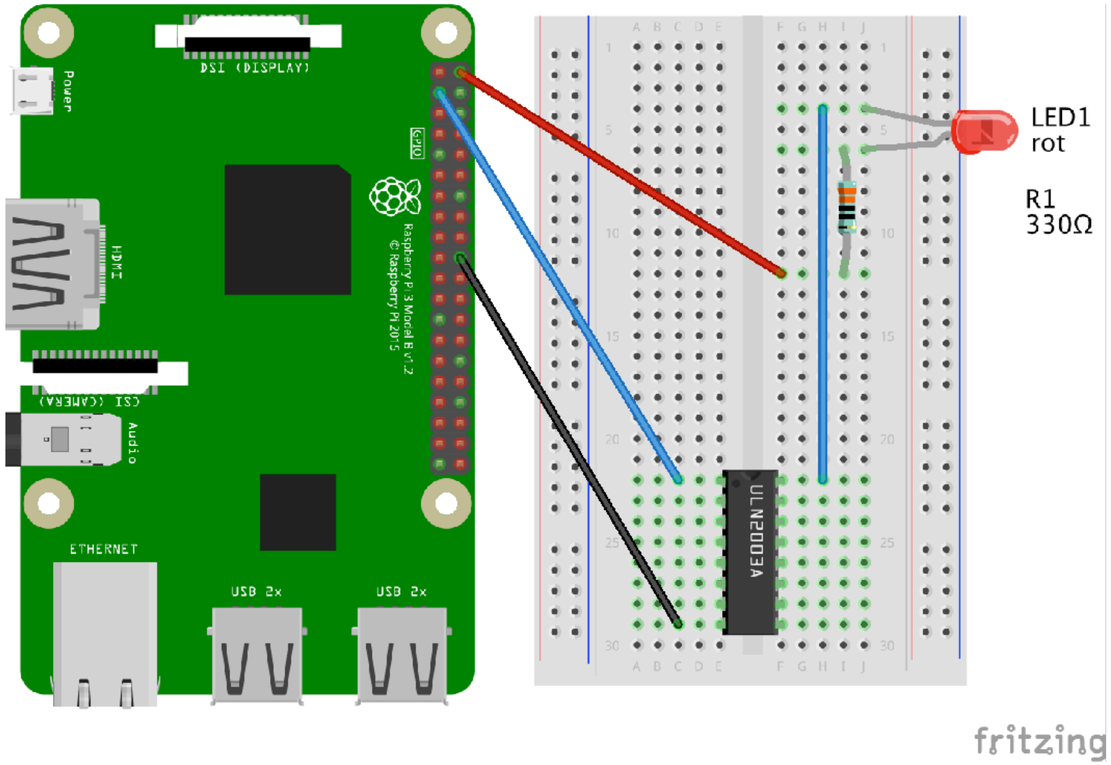
LED über ULN2003A steuern

Ein ULN2003A ist ein integrierter Schaltkreis mit 7 bipolaren NPN-Darlington-Transistoren. Damit kann man eine Spannung bis 50 Volt (V) und einen Strom bis 500 Milliampere (mA) pro Ausgang schalten. Eine Besonderheit ist die bereits integrierte Freilaufdiode an den Ausgängen. Dadurch kann man problemlos Relais, Motoren und andere induktive Lasten schalten.

Einen ULN2003A oder einen ähnlichen Baustein braucht man deshalb, weil man einen einzelnen GPIO und alle GPIOs zusammen nicht mit zu viel Strom belasten darf.

Für eigene Anwendungen und Experimente kann der ULN2003A-Ausgang auch mit eigenen Schaltungen oder Bauteilen beschaltet werden. Bei induktiven Lasten muss man den COM-Anschluss mit der Spannungsversorgung des Bauteils verbinden, um die Freilaufdiode zu aktivieren.

Hinweis: Beachte bitte, dass der ULN2003A gegen Ground (GND) schaltet. Beim Verschalten von Bauteile, die hiervon abweichen kann das Verwirrung stiften.




```
from gpiozero import DigitalOutputDevice
from time import sleep

# Initialisierung von GPIO2 als digitaler Ausgang
led = DigitalOutputDevice(2)

# 1 Sekunde warten / Blinken im Verhältnis 2 zu 3
sleep(1)
led.blink(2,3)
```

Motor über ULN2003A steuern (1)

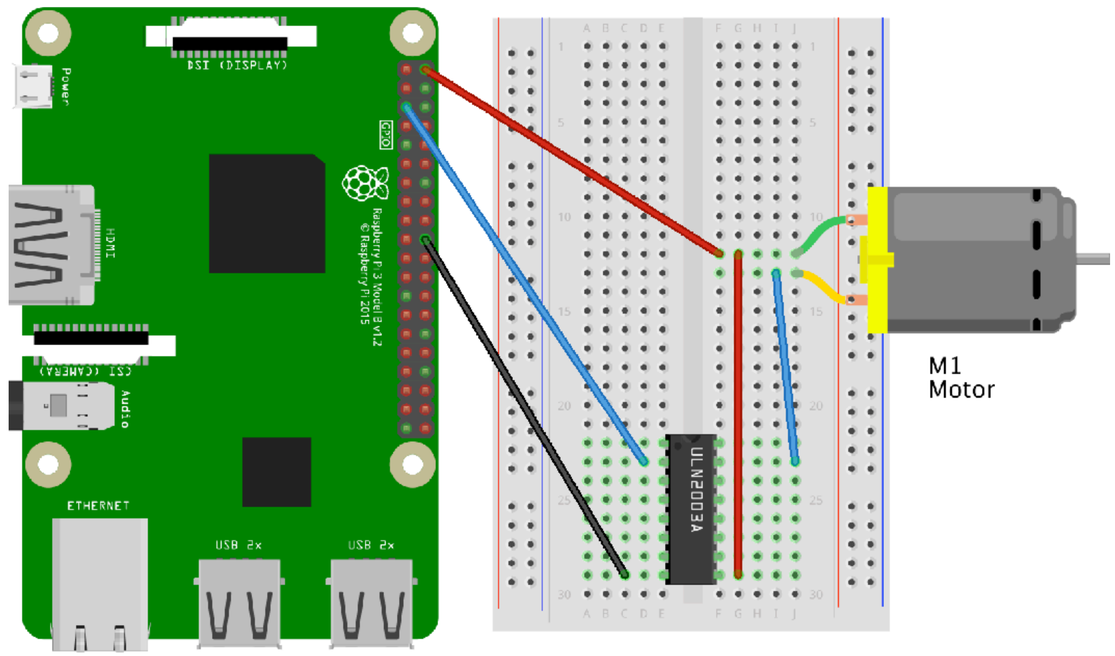
Motoren findet man überall, wo etwas bewegt werden muss. Zum Beispiel Fahrzeuge, Ventilatoren oder Antriebe zum Öffnen und Schließen.

Diese Motoren und Antriebe laufen nicht dauerhaft, sondern nur wenn etwas bewegt werden soll. Zum manuell Bedienen, also zum Einschalten und Ausschalten, eignen sich meist Taster oder Schalter. Zur elektrischen Steuerung gibt es oft Relais, Halteschaltungen, mit und ohne Zeitgeberschaltung.

Zur Steuerung eines Motors kann man eine Treiber-Schaltung mit Transistor und Freilaufdiode verwenden. Oder, was viel einfacher ist, einen ULN2003A-IC. Da ist die Treiber-Schaltung 7-fach in einem Bauteil integriert.

Der folgende Aufbau ist nicht universell, sondern berücksichtigt die Besonderheiten der Ansteuerung des ULN2003A mit einem Motor.

Beachte bitte die Hinweise auf der nachfolgenden Seite!




```
from gpiozero import DigitalOutputDevice
from time import sleep

# Initialisierung von GPIO3 als digitaler Ausgang
motor = DigitalOutputDevice(3)

# Motor: Ein / Warten / Aus
motor.on()
sleep(3)
motor.off()
```

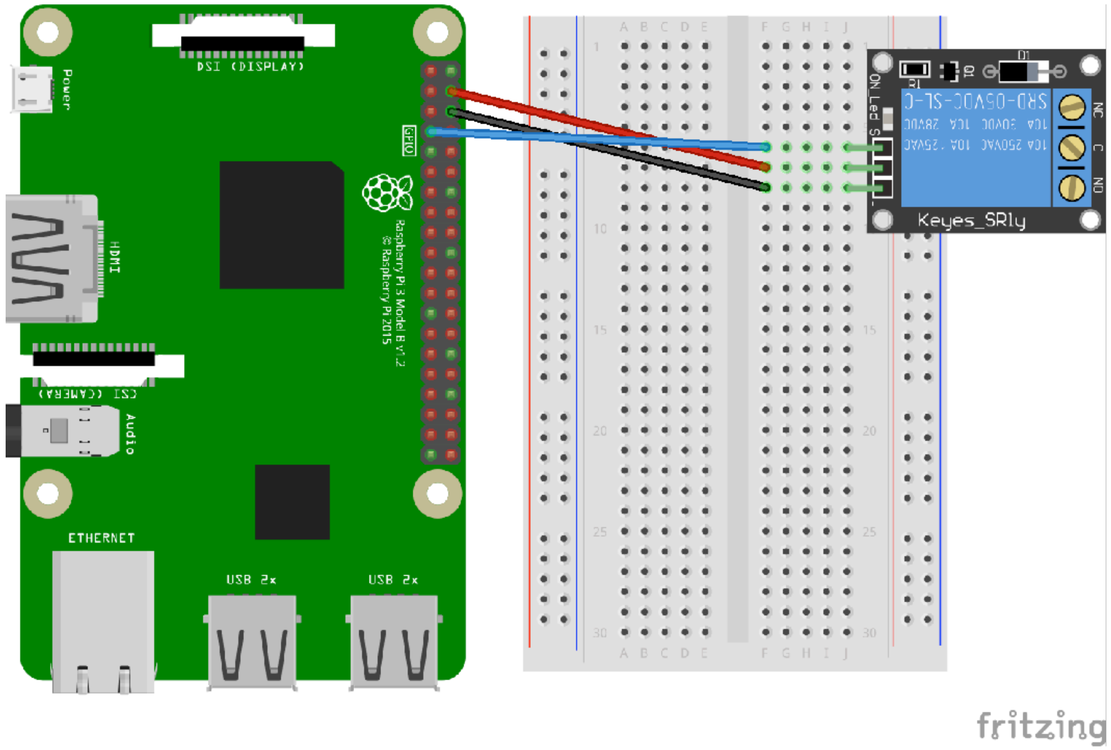
Motor über ULN2003A steuern (2)

- Zur Steuerung eines Motors mit dem Raspberry Pi kann ein GPIO genutzt werden, der als Ausgang programmiert wird. Allerdings darf ein Motor nicht direkt mit dem GPIO-Ausgang verbunden werden. Denn ein GPIO-Ausgang eignet sich nicht zur Stromversorgung des Motors. Ein Verbraucher, wie ein Motor, braucht immer eine eigene Stromversorgung. Zum Beispiel ein Netzteil. Unabhängig davon muss zwischen GPIO-Pin und Verbraucher eine Treiber-Schaltung oder ein potentialfreier Kontakt geschaltet werden, damit der Verbraucher, in dem Fall der Motor, unabhängig vom Raspberry Pi geschaltet werden kann. Wenn der Motor mit 12 Volt läuft, dann funktioniert das mit den 3,3 oder 5 Volt vom Raspberry Pi nicht.
- Beachte bitte, dass der ULN2003A gegen Ground (GND) schaltet. Beim Verschalten von Bauteile, die hiervon abweichen kann das Verwirrung stiften.
- Motoren sind Verbraucher, die einen vergleichsweise hohen Strom verbrauchen. Insbesondere beim Anlaufen. Also im Einschaltmoment. In der aufgebauten Schaltung wird der Motor über die 5-Volt-Leitung vom Raspberry Pi mit Strom versorgt.
- Was wichtig ist, dass man weiß, dass abhängig vom Raspberry-Pi-Model und vom verwendeten Netzteil, der Aufbau funktionieren kann, oder auch nicht. Im günstigsten Fall wird im Desktop rechts oben die Meldung „Low voltage warning. Please check your power supply.“ erscheinen. Das heißt nichts anderes, als dass die Belastung durch den Motor die Versorgung im Raspberry Pi zur Unterspannung führt. Wenn man das nicht abstellt, wird der Raspberry Pi einfach abstürzen. Im schlimmsten Fall wird der Raspberry Pi sofort ausgehen, weil das Netzteil nicht genug Strom liefert.
- Stelle für den Dauerbetrieb von externer Beschaltung eine eigene Stromversorgung bereit (Netzteil). Nur so ist ein problemloser Betrieb möglich.

Relais-Board ohne ULN2003A steuern

Um ein Relais mit einem Raspberry Pi zu schalten benötigt man in der Regel ein eigenes Netzteil für das Relais und zusätzliche Beschaltung in Form eines Transistors, zwei Widerstände und einer Diode. Und dann muss das noch korrekt dimensioniert sein.

Der Vorteil eines Relais-Boards ist, dass die notwendigen Bauteile korrekt dimensioniert und auf der Platine mit dem Relais richtig verschaltet sind. Außerdem hat ein Relais-Board auch noch schraubbare Anschlüsse zum Verbinden nachfolgender Schaltungen oder Geräte.



Zur Steuerung des Relais-Boards mit dem Raspberry Pi kann ein GPIO genutzt werden, der als Ausgang programmiert wird.

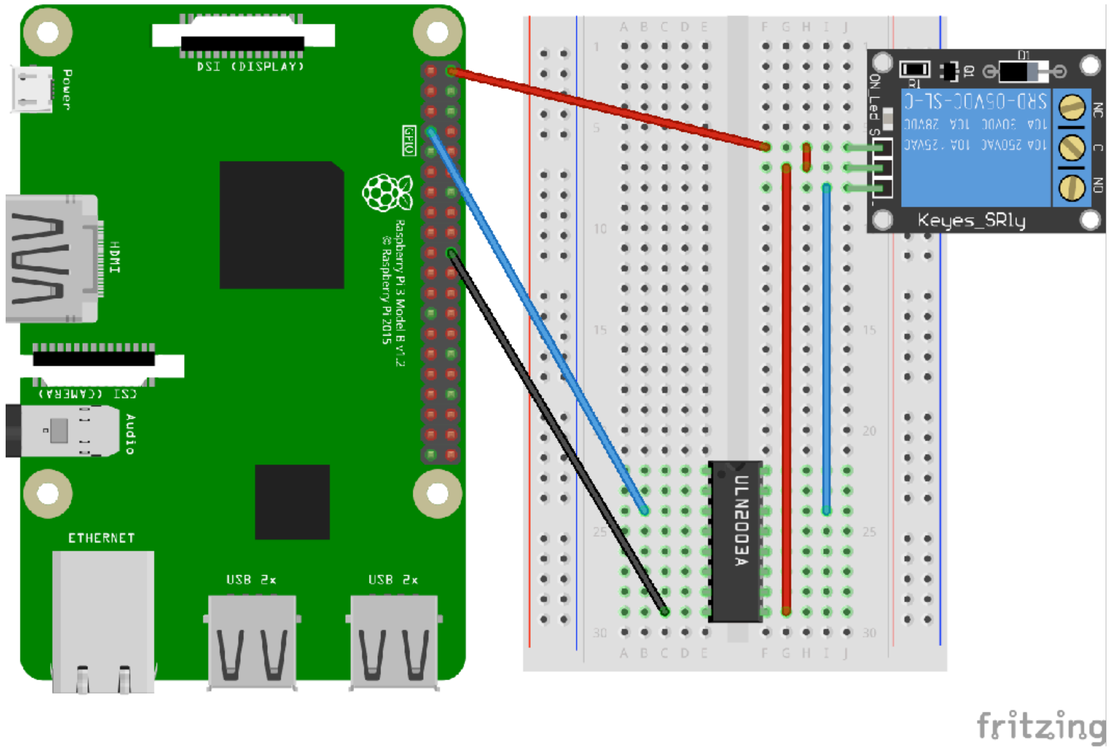
Dieses Relais-Board ist für Mikrocontroller, wie dem Arduino, konzipiert und arbeitet mit TTL-Steuersignalen (5 Volt und 0 Volt). Der Raspberry Pi gibt an seinen GPIO-Ausgängen aber nur eine Spannung von 3,3 Volt aus. Das reicht normalerweise aus, um ein TTL-High-Signal zu erkennen und zu schalten.

```
from gpiozero import DigitalOutputDevice
from time import sleep

# Initialisierung von GPIO4 als digitaler Ausgang
relais = DigitalOutputDevice(4)

# Relais: Ein / Warten / Aus
relais.on()
sleep(3)
relais.off()
```

Relais-Board über ULN2003A steuern

Die typischen Relais-Boards sind für Mikrocontroller, wie dem Arduino, konzipiert und arbeiten eingangsseitig mit TTL-Steuersignalen (5 Volt und 0 Volt). Der Raspberry Pi gibt an seinen GPIO-Ausgängen aber nur eine Spannung von 3,3 Volt aus. Das reicht normalerweise aus, um ein TTL-High-Signal zu erkennen und zu schalten. Die Frage ist, ob das in der Praxis immer so funktionieren wird und betriebssicher ist.

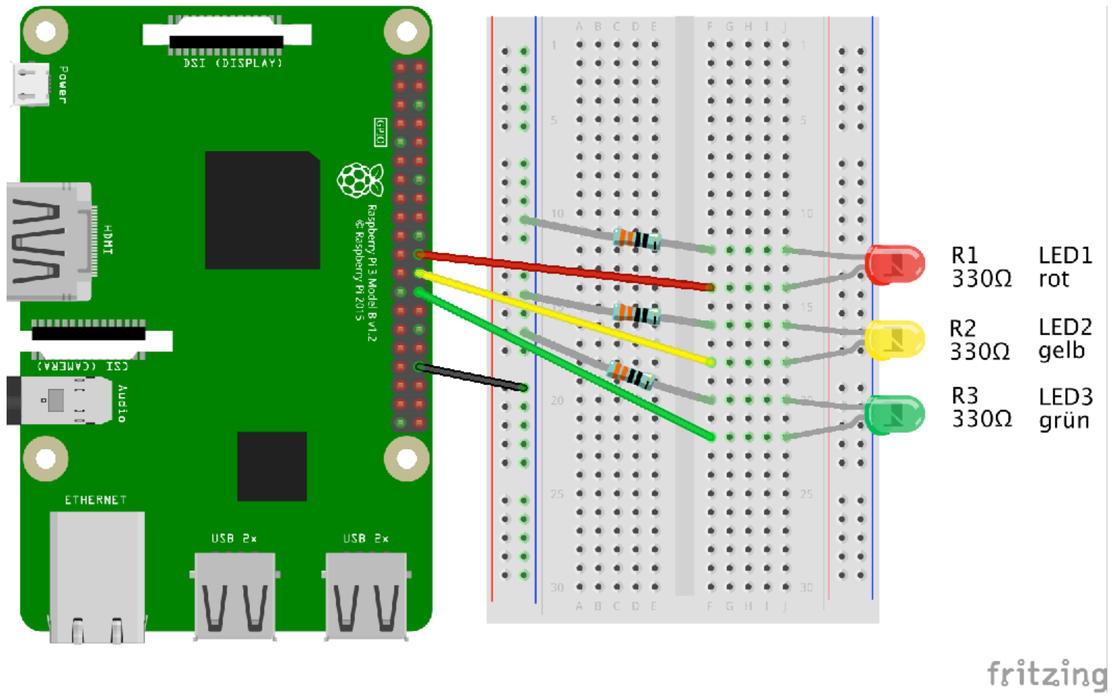
Wenn knapp dimensionierte Bauteile und Baugruppen zu gelegentlicher Fehlfunktion führen, dann kann das ungeahnte Folgen haben. Deshalb verbinden wir das vorhandene Relais-Board nicht direkt mit dem GPIO, sondern schalten ein ULN2003-IC dazwischen. Zwar ist dieses IC auch für TTL-Signale gedacht, also 5 und 0 Volt, allerdings reagiert es wegen der integrierten Darlington-Schaltung ziemlich empfindlich. Das heißt, es steuert sicherer durch als die Eingangsschaltung eines Relais-Boards.




```
from gpiozero import DigitalOutputDevice
from time import sleep

# Initialisierung von GPIO4 als digitaler Ausgang
relais = DigitalOutputDevice(4)

# Relais: Ein / Warten / Aus
relais.on()
sleep(3)
relais.off()
```

LED-Ampel-Steuerung

Eine LED-Ampel-Steuerung lässt mit einigen Zeilen Code in Python mit der Bibliothek GPIO Zero schnell schreiben. GPIO Zero kennt dazu ein eigenes Initialisierungs-Device für eine Ampel-Steuerung. Damit lassen sich drei LEDs zu einer Ampel zusammenfassen und gemeinsam steuern.

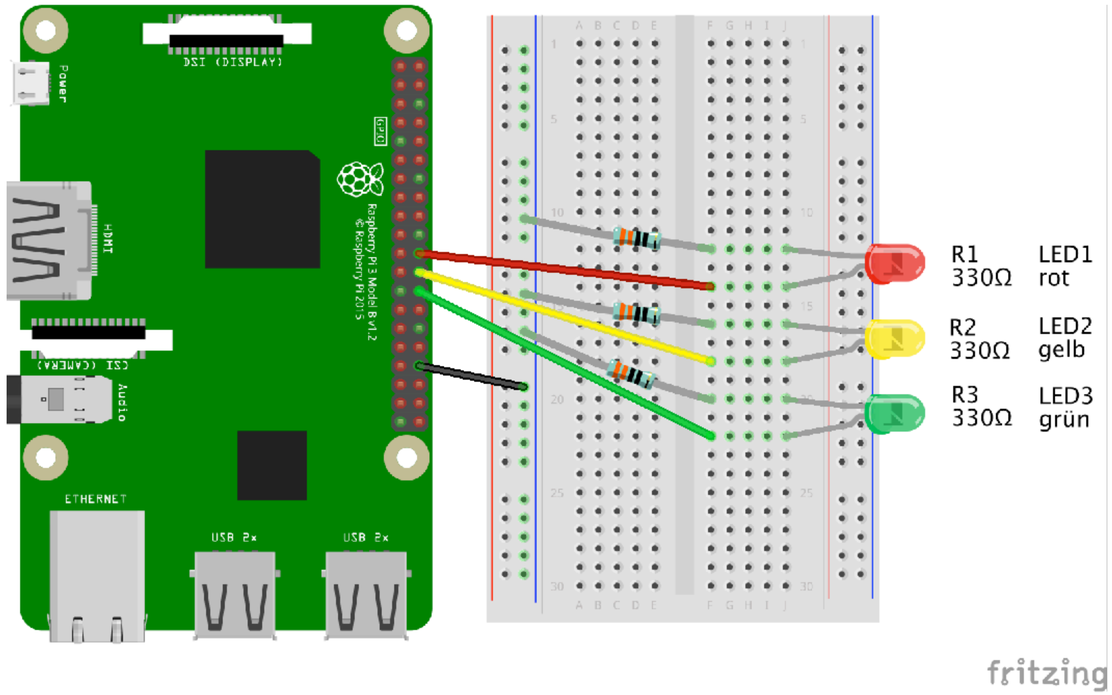



```
from gpiozero import TrafficLights
from time import sleep

# Initialisierung: GPIO25/GPIO8/GPIO7 als Ampel
lights = TrafficLights(25, 8, 7)

# Definition einer Funktion für die Ampelphasen
def traffic_light_sequence():
    # Wiederholung einleiten
    while True:
        # Rot-Phase
        yield (1, 0, 0)
        sleep(7)
        # Rot-Gelb-Phase
        yield (1, 1, 0) # rot+gelb
        sleep(2)
        # Grün-Phase
        yield (0, 0, 1) # grün
        sleep(5)
        # Gelb-Phase
        yield (0, 1, 0) # gelb
        sleep(2)

# Aufruf der Steuerung für die Ampelphasen
lights.source = traffic_light_sequence()
```

LED-Lauflicht

Ein LED-Lauflicht lässt mit einigen Zeilen Code in Python mit der Bibliothek GPIO Zero schnell schreiben. GPIO Zero kennt für eine LED-Reihe ein eigenes Initialisierungs-Device, mit dem sich mehrere LEDs mit einem Kommando gemeinsam steuern und zu als Lauflicht nutzen lassen.

```
from gpiozero import LEDBoard
from time import sleep

# Initialisierung von GPIO25, GPIO8, GPIO7,
GPIO20 und GPIO21 als LED-Reihe (Ausgang)
leds = LEDBoard(25, 8, 7, 20, 21)

# Wiederholung einleiten
while True:
    # LED-Status setzen
    leds.value = (1, 0, 0, 0, 0)
    # Warten (Sekunden)
    sleep(0.2)
    leds.value = (0, 1, 0, 0, 0)
    sleep(0.2)
    leds.value = (0, 0, 1, 0, 0)
    sleep(0.2)
    leds.value = (0, 0, 0, 1, 0)
    sleep(0.2)
    leds.value = (0, 0, 0, 0, 1)
    sleep(0.2)
```



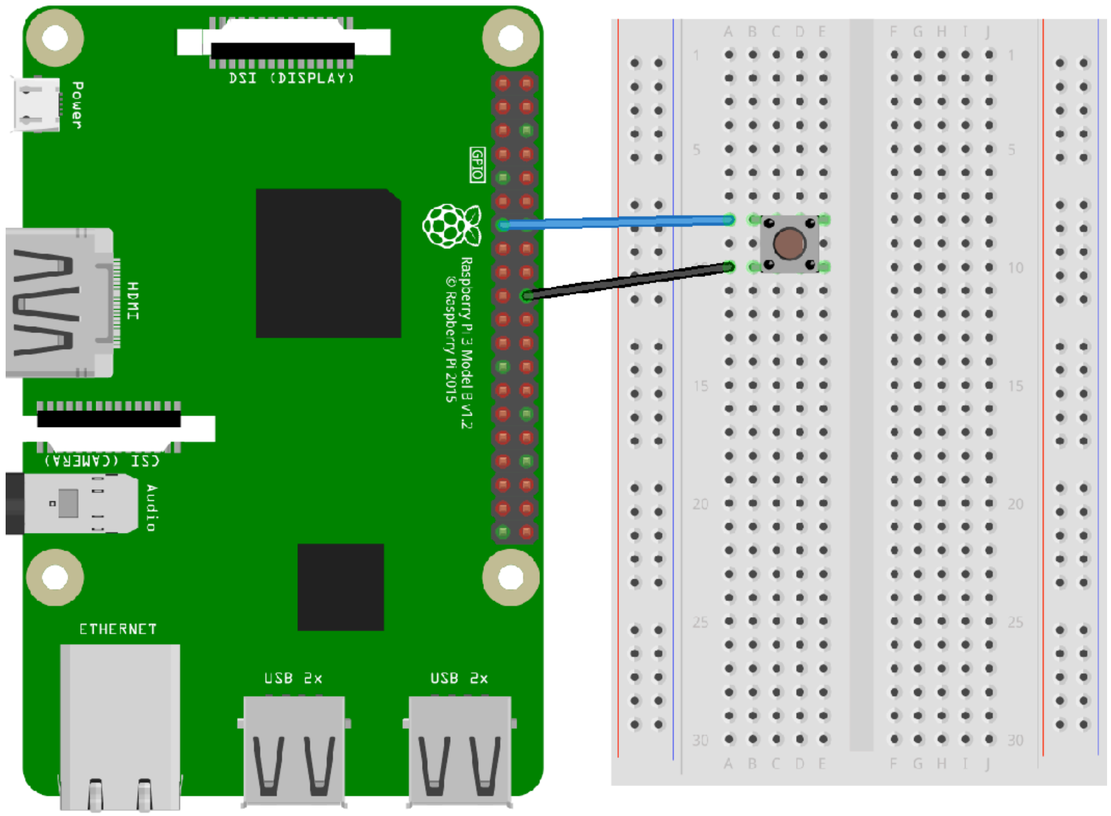
Herunterfahren per Taster

Wie schaltest Du einen Raspberry Pi richtig aus? Genau, Du fährst ihn softwareseitig herunter, auch Shutdown genannt. Wie jeden anderen Computer auch. Dafür gibt es in der Regel im Betriebssystem eine Menü-Funktion.

Durch das softwareseitige Herunterfahren wird sichergestellt, dass alle Speicherinhalte auf den Massenspeicher geschrieben und alle Prozesse kontrolliert beendet werden. Auf diese Weise vermeidest Du Datenverlust und Inkonsistenzen im Speicher, die einen erneuten, erfolgreichen Bootvorgang verhindern würden.

Den Wunsch nach einem „Shutdown“ per Taster oder Schalter hört man öfter. Die meisten Raspberry Pis haben das nicht vorgesehen. Aber Du kannst diese Funktion selber nachrüsten. Dafür brauchst Du nur einen Taster und ein Programm, dass das Drücken des Tasters auswertet und den „Shutdown“ durchführt.

Dabei reicht es nicht, den Taster einfach nur zu drücken. Der Taster muss 2 Sekunden gedrückt gehalten werden, damit eine Funktion ausgeführt wird.



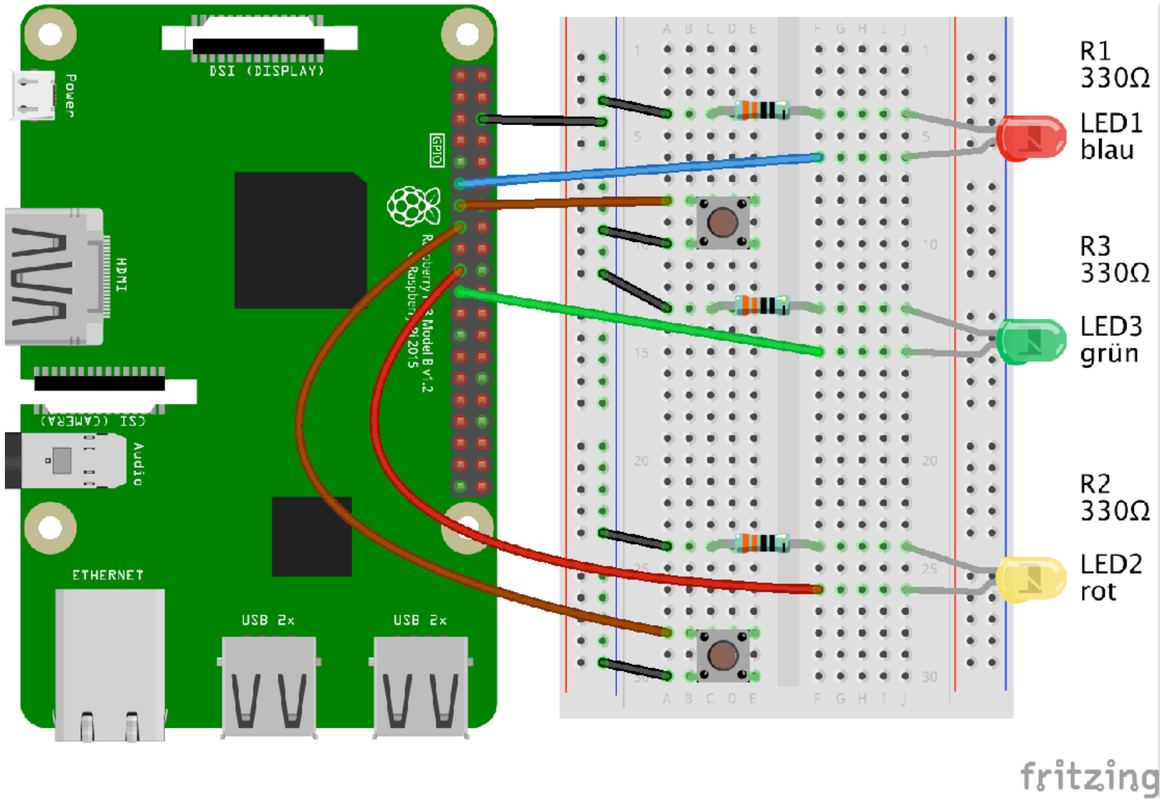
...

Reaktionsspiel mit Tastern und LEDs (1)

Experimentieren mit Tastern und LEDs ist ja ganz nett. Aber soll es das schon gewesen sein? Taster drücken und LED geht an. Voll langweilig. Es wird Zeit das wir etwas wirklich sinnvolles mit Tastern und LEDs machen. Ein Spiel gefällig? Du musst natürlich zu Zweit sein, sonst macht das keinen Spaß.

Na dann los. Aufgebaut und hergeschaut. Wer hat am schnellsten den Taster gedrückt, wenn die grüne LED angeht?

Im folgenden Programm werden zwei Taster und drei Leuchtdioden definiert. Die grüne LED soll nach einer zufälligen Zeit zwischen 5 und 10 Sekunden angehen. Wer zuerst seinen Taster drückt, dessen Leuchtdiode leuchtet und der Spieler hat gewonnen. Wessen LED leuchtet kann das Spiel mit Drücken auf den Taster erneut starten.



Reaktionsspiel mit Tastern und LEDs (2)

```
from gpiozero import Button, LED
from time import sleep
import random

# LEDs initialisieren
led_1 = LED(17)
led_2 = LED(10)
led_3 = LED(9)

# Taster initialisieren
btn_1 = Button(27)
btn_2 = Button(22)

# Definition einer Funktion
def pressed_btn_1():
    if led_1.is_lit:
        start_game()
        return
    if led_3.is_lit:
        led_1.on()
        led_3.off()
```

```
def pressed_btn_2():
    if led_2.is_lit:
        start_game()
        return
    if led_3.is_lit:
        led_2.on()
        led_3.off()

def start_game():
    # Alle LEDs ausschalten
    led_1.off()
    led_2.off()
    led_3.off()
    # Zufällige Zeit zwischen 5 und 10 Sekunden
    time = random.uniform(5, 10)
    sleep(time)
    led_3.on()

# Wenn ein Button gedrückt wird
btn_1.when_pressed = pressed_btn_1
btn_2.when_pressed = pressed_btn_2

# Spiel starten
start_game()
```

Weitere Ideen für Dich

So, jetzt hast Du genug experimentiert. Du bist jetzt bereit eigene Anwendungen zu programmieren. Wenn Du noch nicht weist, was Du jetzt selber bauen sollst, hier noch ein paar Ideen, die natürlich komplexer sind, als das, was du bisher gemacht hast.

Die Lösungen für folgende Ideen kannst Du teilweise aus den vorgegangenen Experimenten ableiten und um eigene Ideen erweitert.

Viel Vergnügen.

1. Schalte den Motor durch Drücken eines Tasters jeweils ein und aus.

2. Starte und Stoppe das Lauflicht durch Drücken eines Tasters.

3. Erweitere die einzelne Ampel um 3 weitere Ampeln für eine Kreuzung und programmiere die Steuerung dafür richtig.

4. Programmiere eine LED-Steuerung so, dass die LED wie eine Kerze flackert.

Lust auf mehr?

Elektronik-Set Sensor Edition

Das Elektronik-Set Sensor Edition ist eine Sammlung beliebter Sensoren und Bauteile für die Hardware-nahe Programmierung mit Mikrocontrollern und Mini-Computern.

Es werden verschiedene Bereiche abgedeckt, wie Temperatur, Bewegungserkennung, Akustikerkennung, Lichterkennung, Bewegen und Lichtsteuerung.

Spannende Experimente und Anwendungen aus dem Bereich Smart Home, Robotik und Automation warten auf Dich.

https://www.elektronik-kompendium.de/shop/elektronik-set/sensor-edition

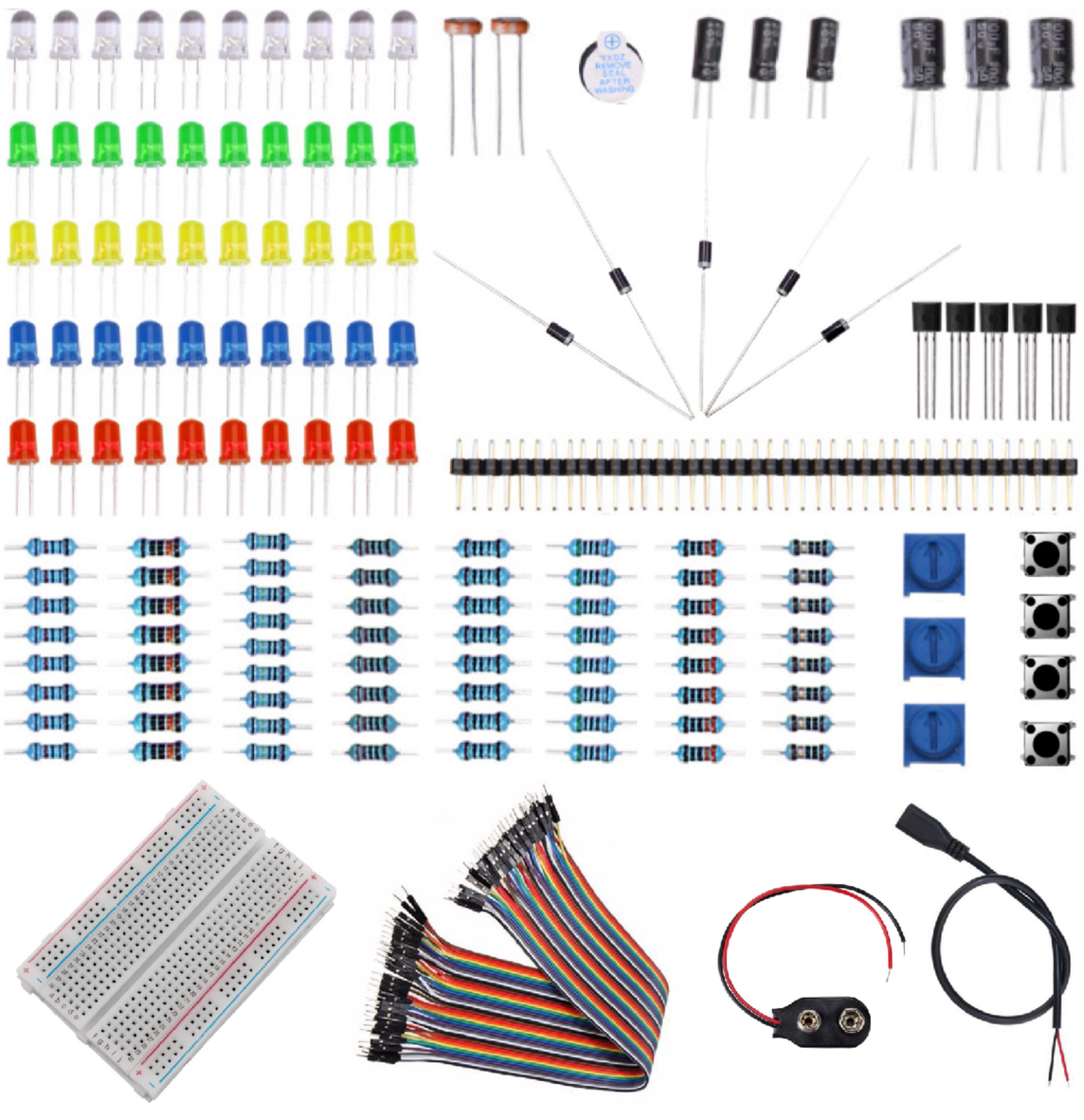
Elektronik-Set Eingabe Ausgabe Edition

Das Elektronik-Set Eingabe Ausgabe Edition ist eine Sammlung beliebter Bauteile zur Dateneingabe, Steuerung und Datenausgabe für die Hardware-nahe Programmierung mit Mikrocontrollern und Mini-Computern.

Wenn Sie bereits das Elektronik-Set Pico Edition oder Pico WLAN Edition haben, dann können Sie es mit diesem Elektronik-Set sinnvoll erweitern und viele weitere spannende Experimente durchführen.

https://www.elektronik-kompendium.de/shop/elektronik-set/eingabe-ausgabe-edition

Elektronik-Set Starter Edition



Mit Elektronik ohne Löten experimentieren

Das Elektronik-Set Starter Edition ist die optimale Ergänzung zum Elektronik-Guide. Das Elektronik-Set enthält alle und noch viel mehr Bauteile, um alle Schaltungen und Experimente nachzubauen.

Zusätzlich enthält das Elektronik-Set:

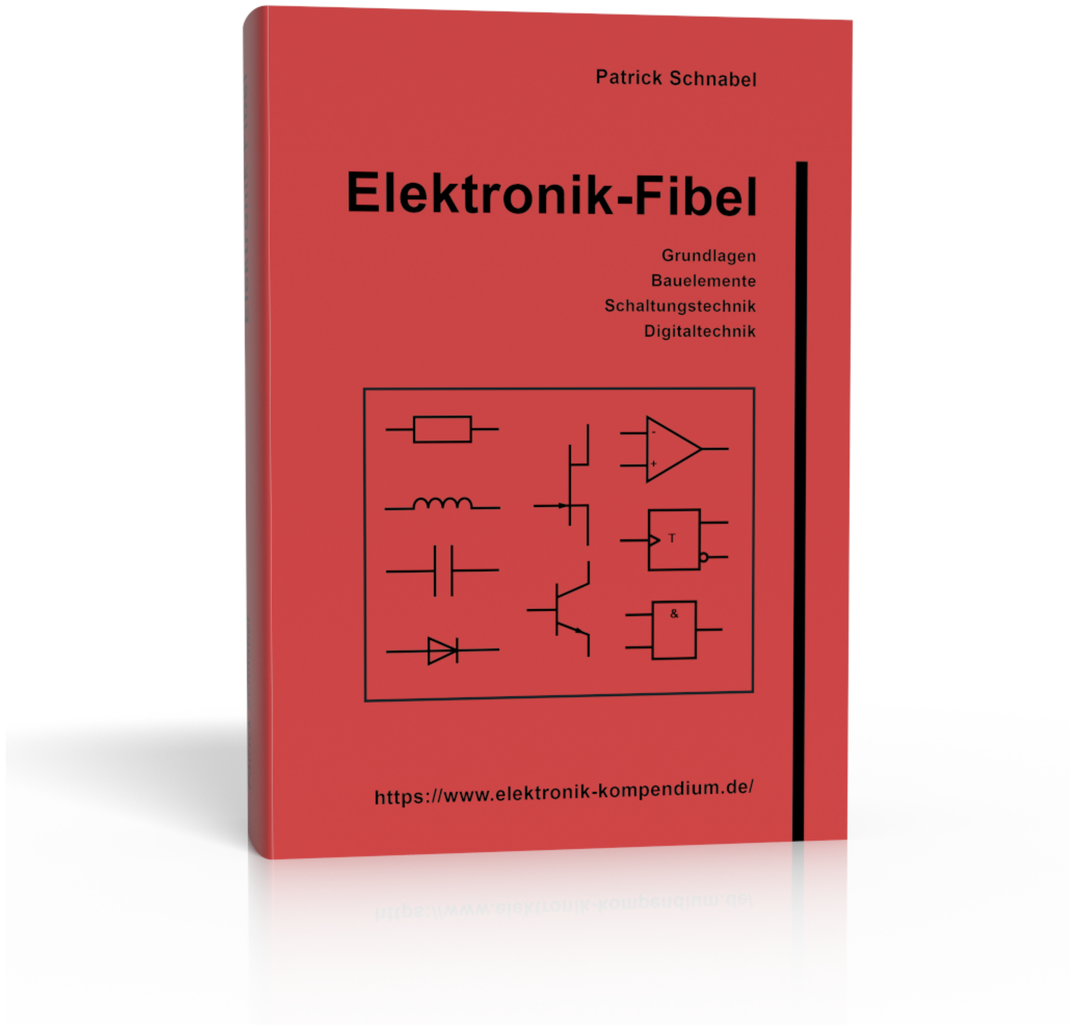
- 1 Steckbrett mit 400 Pins
- 40 Verbindungskabel
- 1 Batterie-Clip für einen 9-Volt-Block
- 1 Micro-USB-Adapter für ein USB-Ladegerät

Nicht im Lieferumfang enthalten und zusätzlich empfohlen:

- 9-Volt-Block-Batterie, USB-Netzteil oder USB-Ladegerät

https://www.elektronik-kompendium.de/shop/elektronik-set/starter-edition

Elektronik-Fibel



Elektronik - einfach und leicht verständlich

Elektronik muss nicht schwer sein. Die Elektronik-Fibel beschreibt die Grundlagen der **Elektronik einfach und leicht verständlich**, so dass der Einstieg in die Elektronik so einfach wie möglich gelingt.

Die Elektronik-Fibel eignet sich besonders **zum Lernen auf Klassenarbeiten, Klausuren und Prüfungen** oder als Nachschlagewerk für die Schule und Ausbildung.

Mit den vielen grafischen Abbildungen, Formeln, Schaltungen und Tabellen dient diese Buch dem Einsteiger und auch dem Profi immer und überall als **unterstützende und nützliche Lektüre**.

https://www.elektronik-kompendium.de/shop/buecher/elektronik-fibel