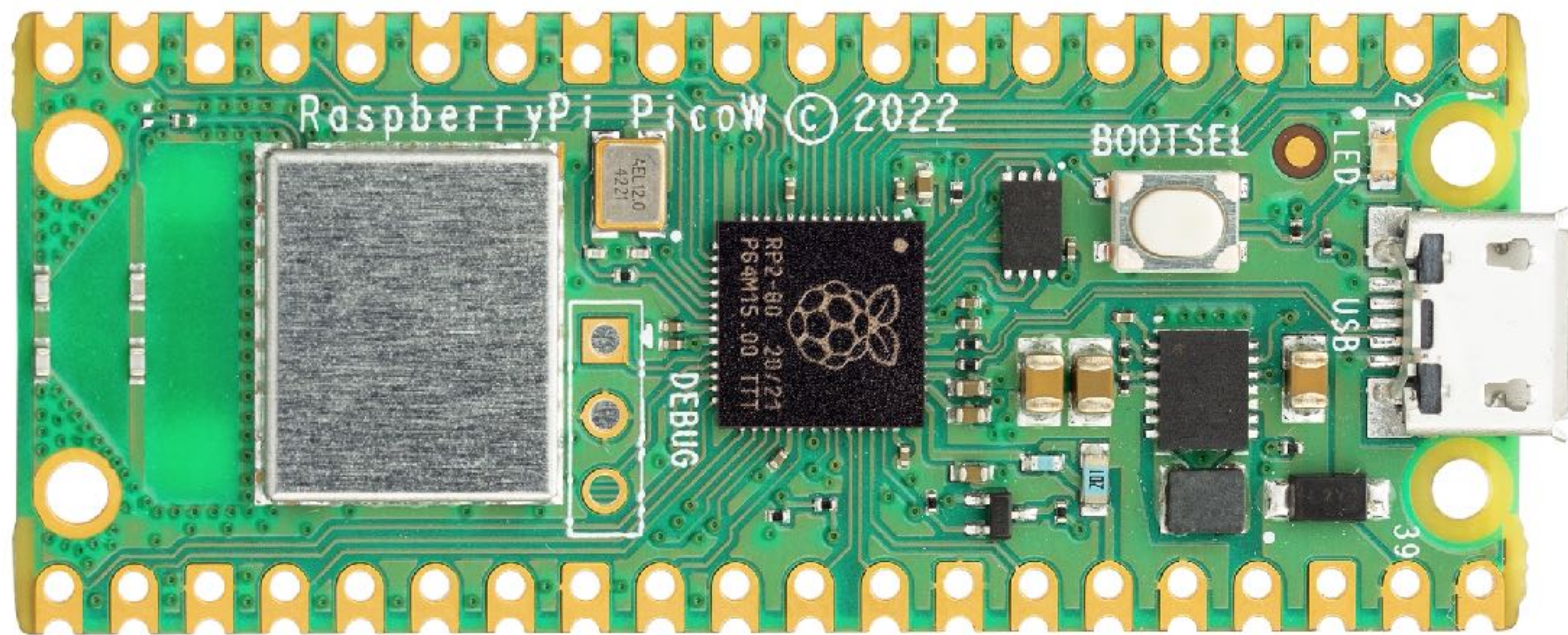


+ MicroPython



Elektronik-Guide Pico WLAN Edition

Impressum

Elektronik-Guide Pico WLAN Edition

Version: 2023-08-04

Herausgeber:

Patrick Schnabel

Droste-Hülshoff-Str. 22/4

71642 Ludwigsburg

Deutschland

USt-ID-Nr.: DE207734730

WEEE-Reg.-Nr.: DE80632679

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



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)

Wenn wir uns die gerade aktuellen technischen Trends anschauen, dann ist irgendwie alles Internet of Things, Smart Home, Cloud Computing oder Künstliche Intelligenz (KI/AI). Wenn wir etwas genauer auf die technischen Details schauen und wie man diese Techniken nutzt, dann steckt in jedem technischen Trend die „Vernetzung“ drin. Es gibt praktisch keinen neuen technischen Trend, der ohne die Kommunikation mit anderen Geräten oder Systemen auskommt.

Während man früher unter Vernetzung das Verbinden von Geräten verstand, stellt durch die Verfügbarkeit von Funknetzen, wie WLAN und Mobilfunk, das Verbinden heute kein Problem mehr dar. Ein Netzwerk ist im Regelfall immer verfügbar.

Heute versteht man unter Vernetzung das Realisieren von Verbindungen auf Anwendungsebene, unabhängig von der Bedienung eines Anwenders. Ziel ist meist das Verknüpfen von Daten, Zuständen und Funktionen, die in Abhängigkeit zueinander stehen und automatisch Geräte steuern und Daten verarbeiten.

Mit einem Mikrocontroller, wie dem Raspberry Pi Pico W kann man nicht nur Hardware-nah Programmieren, sondern die Hardware in ein Netzwerk einbinden und Daten mit anderen Systemen austauschen.

Denkbar wäre, dass Du mit dem Raspberry Pi Pico W eigene drahtlose Sensoren baust, die es in dieser Form noch nicht gibt.

Richtig interessant wird es, wenn Du Dir Anwendungen ausdenkst, mit denen Du räumliche Grenzen überwindest und öffentlich zugängliche Daten abrufen und verarbeiten lässt. Im einfachsten Fall kann das auch nur das Anzeigen von Daten sein.

Beispielsweise:

- Aktuelles Datum und Uhrzeit
- Wetterdaten vom Deutschen Wetterdienst (DWD)
- Gesamtfüllstand der deutschen Gasspeicher

Vorwort (2)

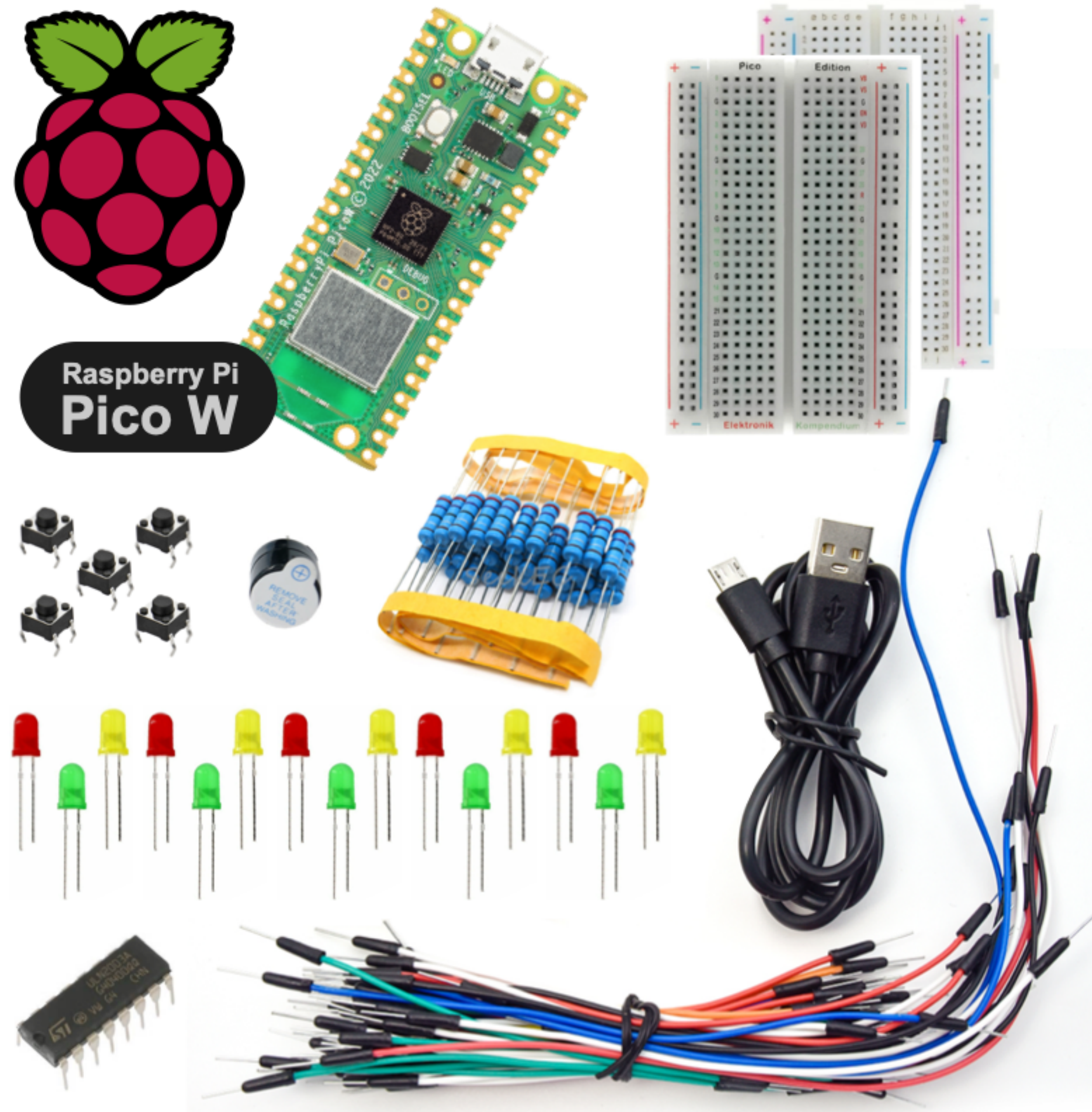
Mit diesen Anregungen zum Ausprobieren wünsche ich Dir viel Spaß.

Patrick Schnabel

PS: So ganz alleine, ohne jede Unterstützung und nur mit einem Raspberry Pi Pico W, macht das nicht wirklich Spaß. Wenn Du Lust hast, in entspannter Atmosphäre mit anderen mit dem Raspberry Pi Pico zu experimentieren, dann sind vielleicht unsere Online-Workshops etwas für Dich.

<https://www.elektronik-kompodium.de/service/events/>

Elektronik-Set Pico WLAN Edition



Hardware-nahes Programmieren mit dem Mikrocontroller Raspberry Pi Pico W und MicroPython.

- Raspberry Pi Pico W mit gelöteten Stiftleisten
- Spezielles Steckbrett mit GPIO-Beschriftung
- Einführung ins Hardware-nahe Programmieren
- Schwerpunkte: WLAN, MQTT und Internet
- Deutschsprachige Anleitung als PDF-Datei zum Download

<https://www.elektronik-kompendium.de/shop/elektronik-set/pico-wlan-edition>

Inhaltsverzeichnis



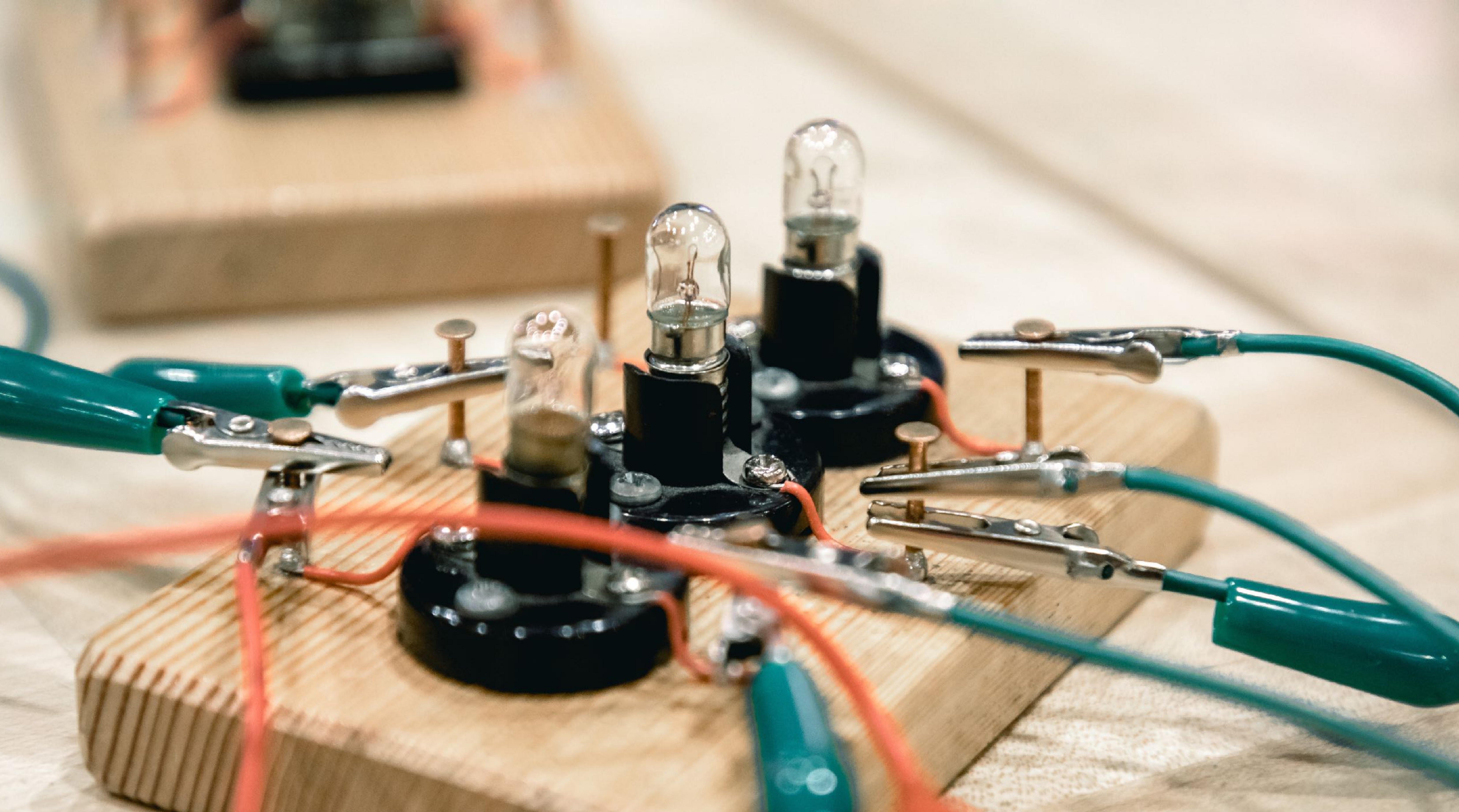
Einführung und Grundlagen



Bauteile

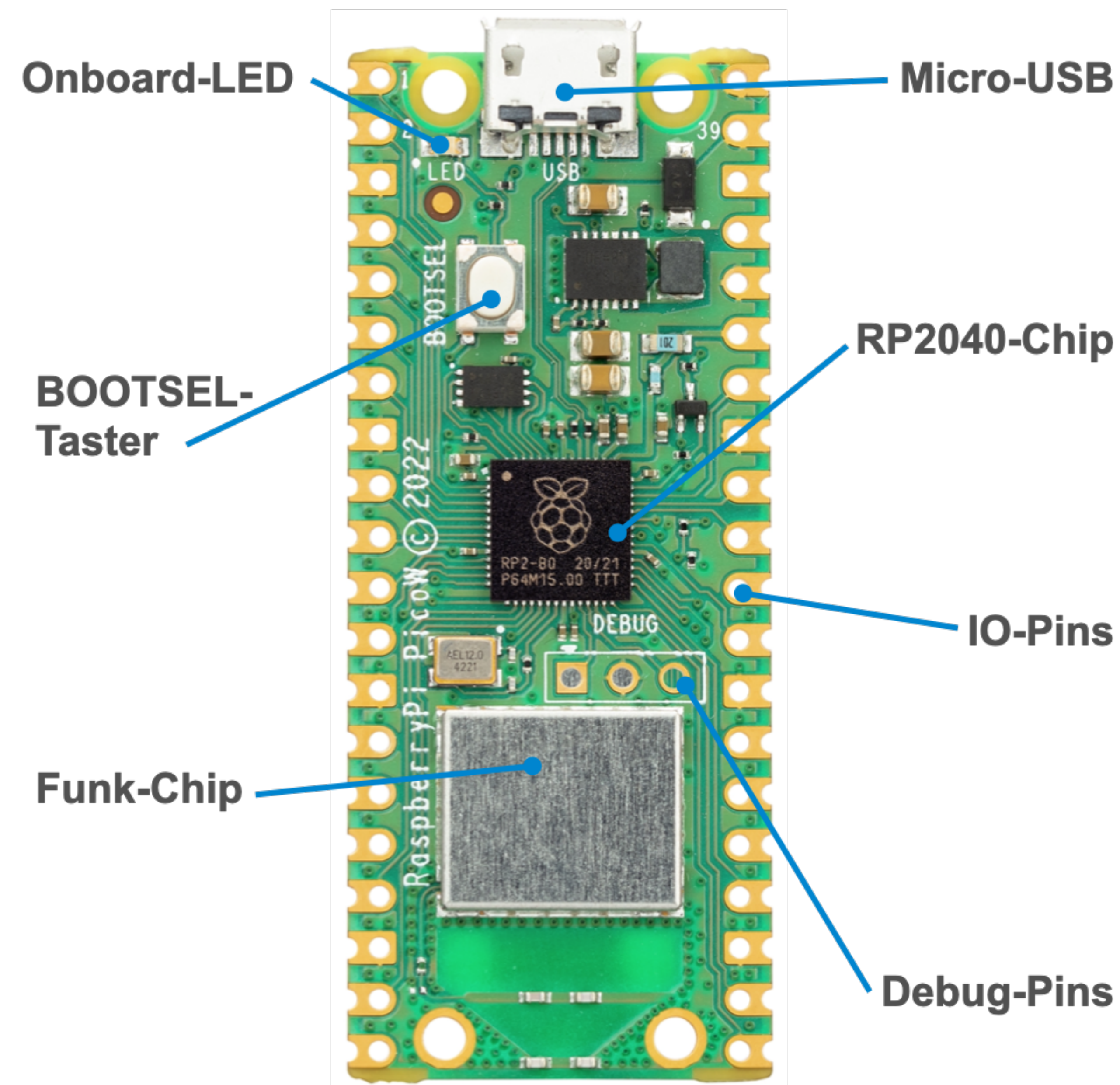


Experimente und Anwendungen



Einführung und Grundlagen

Raspberry Pi Pico: Was ist das?

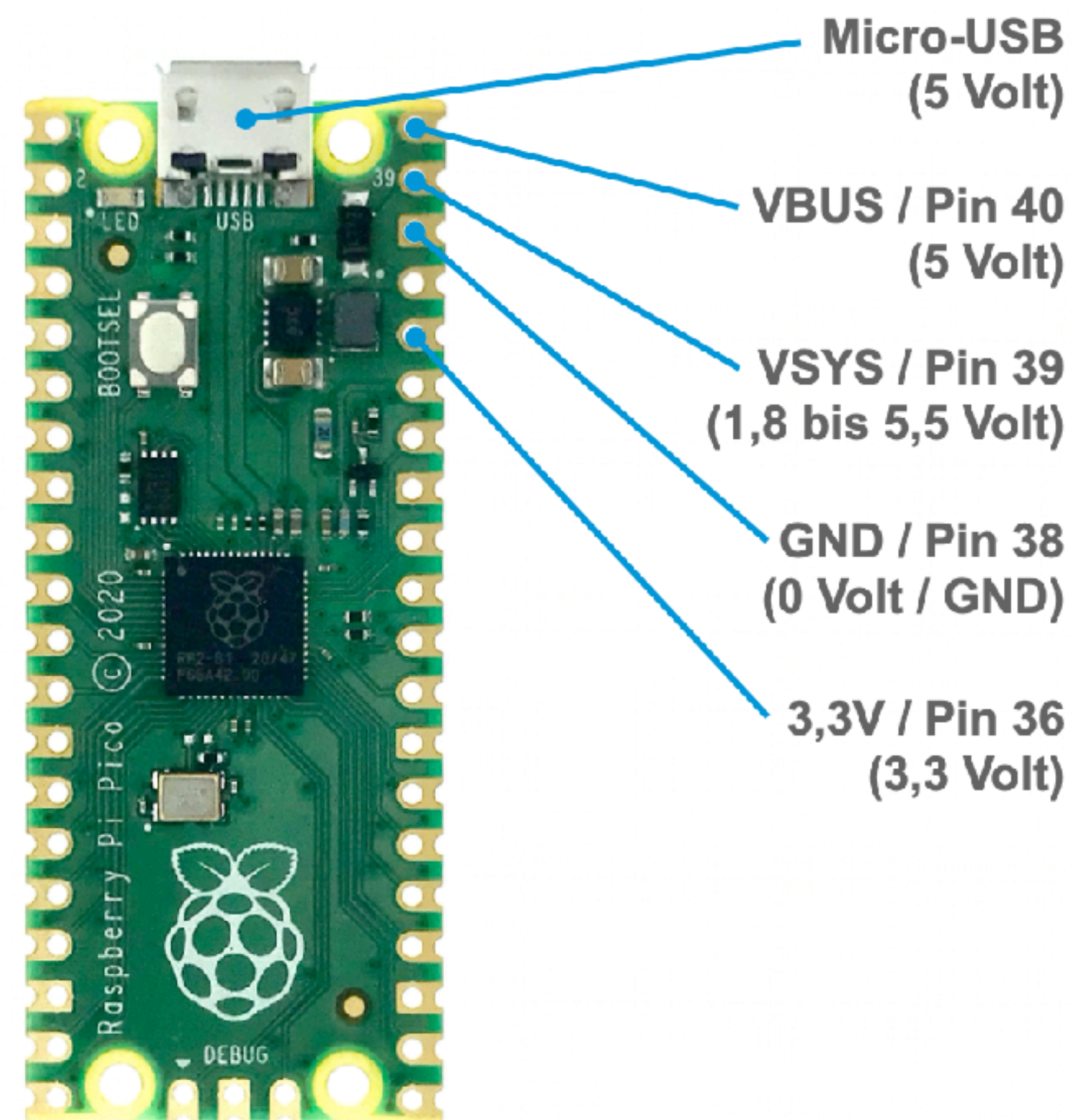


Ein Raspberry Pi Pico ist ein Mikrocontroller-Board von der Raspberry Pi Foundation. Oft wird er liebevoll einfach nur „Pico“ genannt. Er hat einige nützliche Funktionen für Steuerungen, wie man es von anderen Mikrocontrollern kennt. Er lässt sich mit MicroPython, C/C++ und Visual Studio Code programmieren. Die Platine hat das Arduino-Nano-Format und lässt sich über eine Micro-USB-Buchse mit Strom versorgen und programmieren. Der Mikrocontroller ist ein RP2040. Zum Speichern des Programmcodes ist ein 2 MByte großer Flash-Speicher vorhanden. Eine Besonderheit, im Vergleich zu anderen Mikrocontrollern, sind die Programmable IOs (PIO), die unabhängig programmierbar sind.

Ein Mikrocontroller-Board, wie der Pico, ist dafür gedacht, Steuerungen, Regelungen und Automatisierungen zu übernehmen. Dafür verfügt er über zahlreiche analoge und digitale Eingänge und Ausgänge mit unterschiedlichen Funktionen.

Der Raspberry Pi Pico W ist eine Variante, die zusätzlich einen Funk-Chip mit WLAN-Fähigkeit auf dem Board hat.

Energieversorgung



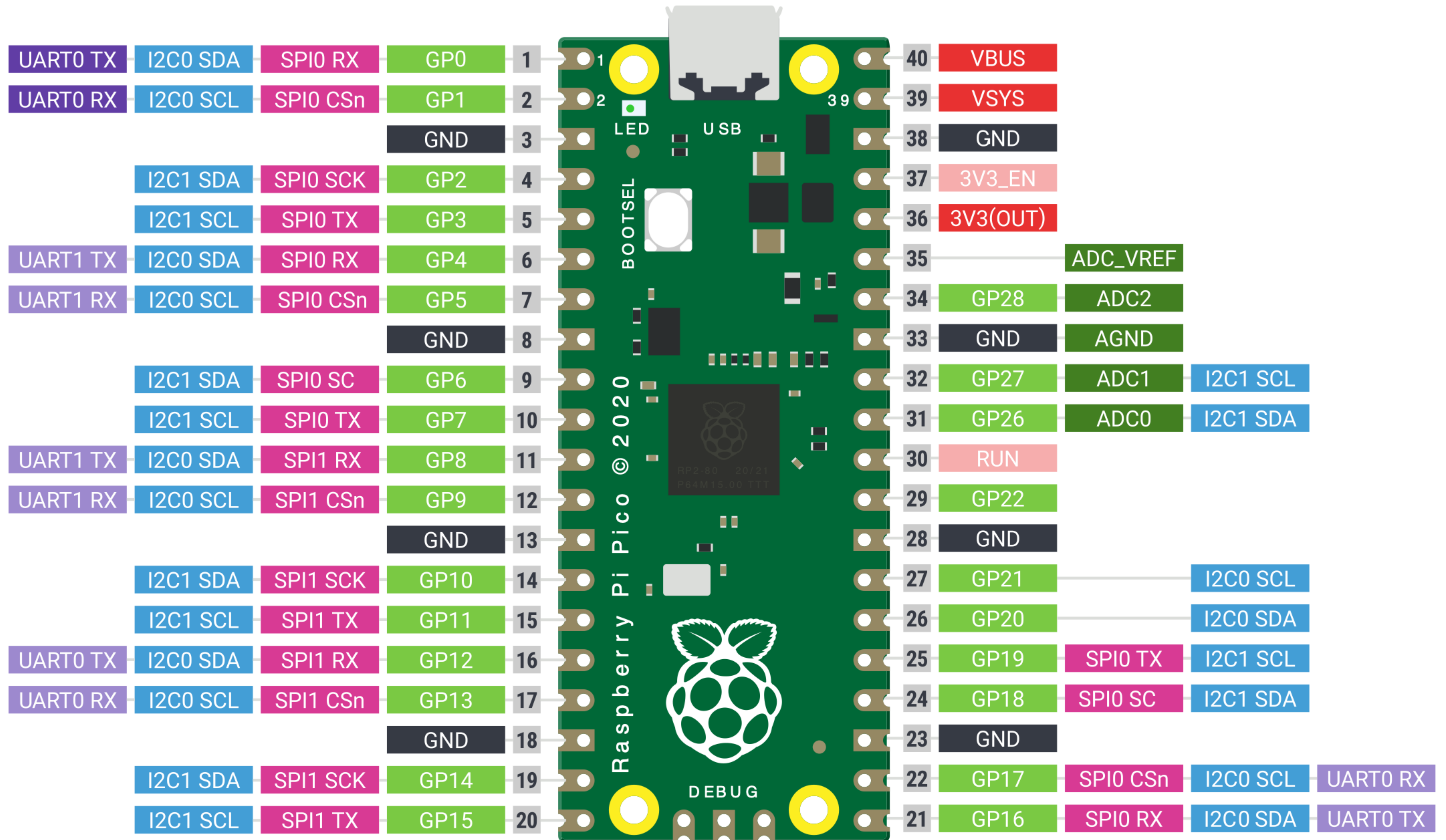
- 5 Volt über den Micro-USB-Anschluss
- 5 Volt über den VBUS-Pin (Pin 40), ohne USB-Speisung oder wenn Pico im USB-Host-Betrieb
- 3,3 Volt über den 3V3-Pin (Pin 36)
- 1,8 bis 5 Volt über den VSYS-Pin (Pin 39)
- Der Anschluss von 0 Volt bzw. GND kann über einen freien GND-Pin erfolgen. Zum Beispiel Pin 38.

Ein Raspberry Pi Pico kann auf mehrere Arten mit Strom versorgt werden. Die einfachste Methode ist, den Micro-USB-Anschluss über ein USB-Kabel mit einem Computer zu verbinden. Alternativ besteht die Möglichkeit im Stand-alone-Betrieb, also ohne Computer, ein normales USB-Netzteil zu verwenden.

Weitere Methoden bestehen im direkten Anschluss einer externen, festen Spannung über den 3,3V-Pin der IO-Pins. Oder, wenn eine andere Spannung verwendet werden muss, dann kann man diese Spannungsquelle mit dem VSYS-Pin der IO-Pins verbinden. Hier darf die Spannung zwischen 1,8 und 5,5 Volt liegen. Auf der Platine ist ein Spannungswandler, der aus einer Eingangsspannung einer Versorgung mit +3,3 Volt macht. Das ist praktisch, wenn man mit 2 oder 3 AA-Batterien eine mobile Energieversorgung realisieren will.

Der Stromverbrauch ist abhängig von den angeschlossenen Bauteilen und liegt bei einfachen Experimenten mit ein paar Leuchtdioden bei maximal 30 mA. Wenn man einen verfügbaren Onboard-Funk-Chip verwendet, dann sollte man mit 100 mA in der Spitze kalkulieren.

GPIO-Belegung (Pinout)



WLAN-Grundlagen

WLAN 1 / Wi-Fi 1: IEEE 802.11 (1999)
WLAN 2 / Wi-Fi 2: IEEE 802.11b (1999)
WLAN 3 / Wi-Fi 3: IEEE 802.11g (2003)
WLAN 4 / Wi-Fi 4: IEEE 802.11n (2009)
WLAN 5 / Wi-Fi 5: IEEE 802.11ac (2014)
WLAN 6 / Wi-Fi 6: IEEE 802.11ax (2021)
WLAN 6E / Wi-Fi 6E: IEEE 802.11ax bei 6 GHz
WLAN 7 / Wi-Fi 7: IEEE 802.11be (?)

WLAN bzw. Wireless Local Area Network ist der Oberbegriff für alle Techniken und Standards mit denen sich lokale Funknetzwerke betreiben lassen. Allerdings wird im allgemeinen Sprachgebrauch der Begriff "WLAN" für ein Funknetzwerk verwendet, dass auf dem Standard IEEE 802.11 basiert.

Die WLAN-Technik ist die einfachste und komfortabelste Möglichkeit, kleine oder mobile Geräte mit einem Netzwerk oder dem Internet zu verbinden.

Der im Raspberry Pi Pico W eingebaute Infineon-Chip CYW43439 unterstützt den WLAN-Standard IEEE 802.11n. Es ist ein relativ alter Standard. Er bildet den kleinsten gemeinsamen Nenner, was auch die Unterstützung in den üblichen Heim-Routern betrifft. Im Datenblatt des Funkchips CYW43439 von Infineon wird eine maximale Empfangsgeschwindigkeit von 65 MBPS (Megabit pro Sekunde) angegeben. Darüberhinaus ist mit MCS7 (Modulation and coding schemes) 72,2 MBit/s erreichbar. Diese Übertragungsraten entsprechen dem rechnerischen Maximum unter Einbeziehung aller Leistungsmerkmale, die im jeweiligen Standard vorgesehen sind. In der Praxis gibt es allerdings Einschränkungen, wegen denen diese Übertragungsraten nicht realisierbar sind. In der Praxis werden die meisten Anwendungen mit einstelligen MBPS-Werten auskommen müssen, was in den meisten Fällen mehr als ausreichend sein dürfte.

WLAN-Troubleshooting (1)

Wenn es mit dem Raspberry Pi Pico W und der WLAN-Verbindung Probleme gibt, dann ist das nicht immer ganz einfach zu lösen. Dazu muss man wissen, dass Funkverbindungen selten problemlos sind, was in der Natur der Sache liegt. Der Umgang mit diesen Problemen ist immer etwas schwierig, weil man nicht „sieht“ was passiert oder wo genau das Problem liegt.

Wir müssen uns, wenn wir nicht genau wissen was los ist, irgendwie dem Problem nähern. Dazu sollten folgende grundsätzlichen Fehlerquellen beachtet und berücksichtigt werden.

- Reichweite der Funksignale
- Frequenzbereich
- Konfiguration des WLANs

Reichweite der Funksignale

Die mögliche Distanz zum Wireless Access Point (WLAN-Router oder ähnlich) wird mit 30 Metern in Innenräumen in der Regel sehr optimistisch angegeben. Doch gerade was Entfernungen bei Funkverbindungen angeht, sollte man vorsichtig sein. Was heute noch geht, muss morgen nicht mehr funktionieren.

Da jeder überall ein WLAN betreiben kann, ist auch überall mit fremden WLANs zu rechnen, über die man keine Kontrolle hat. Fremde WLANs muss man als störende Elemente betrachten, die nicht nur Einfluss auf die Datenrate haben, sondern auch auf die Verbindungsqualität. Mit steigender Anzahl von fremden WLANs und WLAN-Clients in der Umgebung steigt auch die Fehlerrate im eigenen WLAN an.

Je näher sich ein WLAN-Client beim Access Point befindet, desto stabiler wird die WLAN-Verbindung sein. Je weiter weg, desto mehr fremde WLANs werden in die eigene Funkverbindung hineinfunkeln und vielleicht die Verbindung stören.

WLAN-Troubleshooting (2)

Frequenzbereich

Für die WLAN-Technik gibt es mehrere Frequenzbereiche, die je nach Land unterschiedlich breit sein können. Der Hauptfrequenzbereich liegt bei 2,4 GHz und zwei weitere liegen bei 5 GHz und 6 GHz. Der Raspberry Pi Pico W beherrscht nur den Frequenzbereich bei 2,4 GHz. Alles was darüber ist, kann der Pico nicht sehen und demnach auch keine Verbindung aufbauen.

Hinweis: In der Regel braucht jeder Frequenzbereich seine eigene Antenne. Der Pico hat er nur eine Antenne für 2,4 GHz. Für mehr ist auf der Platine einfach kein Platz.

Problem: WLAN-Verbindung wird nicht hergestellt

Folgendes Szenario: In einem MicroPython-Programmcode wird eine Verbindung zu einem WLAN hergestellt. Dazu wurde im Programmcode WLAN-Name (SSID) und WLAN-Passwort eingetragen. Doch leider wird die Verbindung nicht hergestellt.

- Die Verbindung zum WLAN funktioniert nicht.
- Die Verbindung zum WLAN wird nur manchmal hergestellt.
- Die Verbindung zum WLAN funktioniert meistens nicht.

Bevor Du Dich auf die Fehlersuche und Analyse begibst, solltest Du die zwei folgenden Punkte berücksichtigen:

- Hat sich an den Zugangsdaten wirklich nichts geändert? Das andere Clients Probleme mit der WLAN-Verbindung haben, sollte man ausschließen können.
- Hat sich die Position des WLAN-Routers bzw. Access Points nicht verändert? Verändert man die Platzierung des WLAN-Routers, dann verändert sich auch die Reichweite des WLANs. Es kann WLAN-Clients geben, die sich dann nicht mehr verbinden können.

WLAN-Troubleshooting (3)

Analyse: WLAN-Verbindungsstatus ermitteln

Der Treiber für den WLAN-Funkchip gibt mit dem Kommando "wlan.status()" den Status für eine WLAN-Verbindung zurück. Hierzu sind im Treiber mehrere Werte hinterlegt (value of cyw43_wifi_link_status). Die Kürzel und Werte sind allerdings nicht sehr aussagekräftig. Wir klären, was der Status jeweils bedeutet. Außerdem klären wir, was die Lösung sein könnte, wenn eine Verbindung zum WLAN fehlschlägt.

- CYW43_LINK_DOWN (0): Es besteht keine WLAN-Verbindung (Grundzustand).
- CYW43_LINK_JOIN (1): Es besteht ein Verbindungsaufbau-Versuch, der aber noch nicht vollständig abgeschlossen ist.
- CYW43_LINK_NOIP (2): Der Verbindungsaufbau war erfolgreich, das Interface hat aber noch keine IP-Konfiguration. Es kann sein, dass zu diesem Zeitpunkt der Verbindungsaufbau noch nicht vollständig durchlaufen ist, oder das Netzwerk keinen DHCP-Server zum Verteilen einer IP-Konfiguration hat.
- CYW43_LINK_UP (3): Der Verbindungsaufbau war erfolgreich und ist in der Regel vollständig.
- CYW43_LINK_FAIL (-1): Der Verbindungsaufbau wurde abgebrochen.
- CYW43_LINK_NONET (-2): Der Verbindungsaufbau wurde abgebrochen, weil das angegebene WLAN (SSID) nicht gefunden wurde, bzw. nicht erreichbar ist. Hier sollte man einen WLAN-Scan durchführen, um zu prüfen, ob das WLAN für den Raspberry Pi Pico W überhaupt empfangbar ist. Achtung, es kann sein, dass das WLAN-Signal für den Pico zu schwach ist.
- CYW43_LINK_BADAUTH (-3): Der Verbindungsaufbau wurde abgebrochen, weil die Authentifizierung fehlgeschlagen ist. In der Regel wird hier das WLAN-Passwort falsch sein. Hier ist die Eingabe des WLAN-Passworts zu prüfen und zu korrigieren.

WLAN-Troubleshooting (4)

Lösungen bei WLAN-Status -3 und -2

Beim WLAN-Status -3 ist die Authentifizierung fehlgeschlagen. In der Regel ist das eingegebene WLAN-Passwort falsch. Hier ist zu beachten, dass es kein Tippfehler sein muss. Es kann auch an einem Sonderzeichen im Passwort liegen, das nicht richtig interpretiert wird. Bei WLAN-Passwörtern gilt, sich möglichst auf Buchstaben und Zahlen zu beschränken. Das Passwort muss mindestens 8 Zeichen lang sein. Je länger, desto länger dauert das Erraten und desto sicherer ist das Passwort.

Beim WLAN-Status -2 ist das betreffende WLAN nicht zu erreichen. Zuerst sollte man die Schreibweise des WLAN-Namens (SSID) prüfen. Wenn das korrekt ist, dann sollte man einen WLAN-Scan durchführen und prüfen, ob der Pico überhaupt in der Lage ist, das WLAN zu empfangen.

Analyse: WLANs in der Umgebung scannen

Ausgangspunkt bei unklaren WLAN-Problemen sollte ein WLAN-Scan sein, der die empfangbaren WLANs in der Umgebung anzeigt.

```
# Bibliotheken laden
import network

# Client-Betrieb
wlan = network.WLAN(network.STA_IF)

# WLAN-Interface aktivieren
wlan.active(True)

# WLANs ausgeben
print(wlan.scan())
```

Hier sollte das WLAN, mit dem sich der Raspberry Pi Pico W verbinden soll, aufgelistet sein. Wenn das nicht der Fall ist, dann sollte man die Verbindung noch mit einem anderen WLAN-Client prüfen. Wenn das WLAN mit einem anderen WLAN-Client funktioniert, dann wird das Problem nicht der Programmcode oder der Pico sein. Die Lösung ist dann beim WLAN-Router bzw. Access-Point zu suchen.

WLAN-Troubleshooting (5)

Analyse: WLAN-Router oder Access-Point

Wenn die WLAN-Verbindung nur manchmal funktionieren, dann wird es vermutlich nicht an falschen Zugangsdaten liegen. Und sicherlich auch nicht an einem fehlerhaften Programmcode. Du wirst den Fehler woanders suchen müssen. Vermutlich stimmt etwas mit der Konfiguration des WLAN-Routers oder des Access-Points nicht.

In manchen WLAN-Basisstationen existiert ein Kompatibilitätsmodus, der auch alte WLAN-Standards und somit alte WLAN-Geräte unterstützt. Nun würde man bei einem Raspberry Pi Pico W nicht an ein altes WLAN-Gerät denken. Doch wenn man in das Datenblatt des Pico W oder des Funkchips nachschaut, dann wird man feststellen, dass dort explizit die Übertragungsmodi der beiden alten WLAN-Standards IEEE 802.11b und 11g erwähnt werden. Wenn der Pico W keine WLAN-Verbindung mit dem Standard IEEE 802.11n hinbekommt, weil dafür das Funksignal zu schwach ist, dann muss der Funkchip des Pico W zwangsläufig auf die Übertragungsmodi von 11g oder sogar 11b runterschalten. Aber, wenn der Access-Point das nicht anbietet, dann sieht der Funkchip des Pico W dieses WLAN nicht und kann sich auch nicht verbinden.

Lösung: Konfiguration WLAN-Router

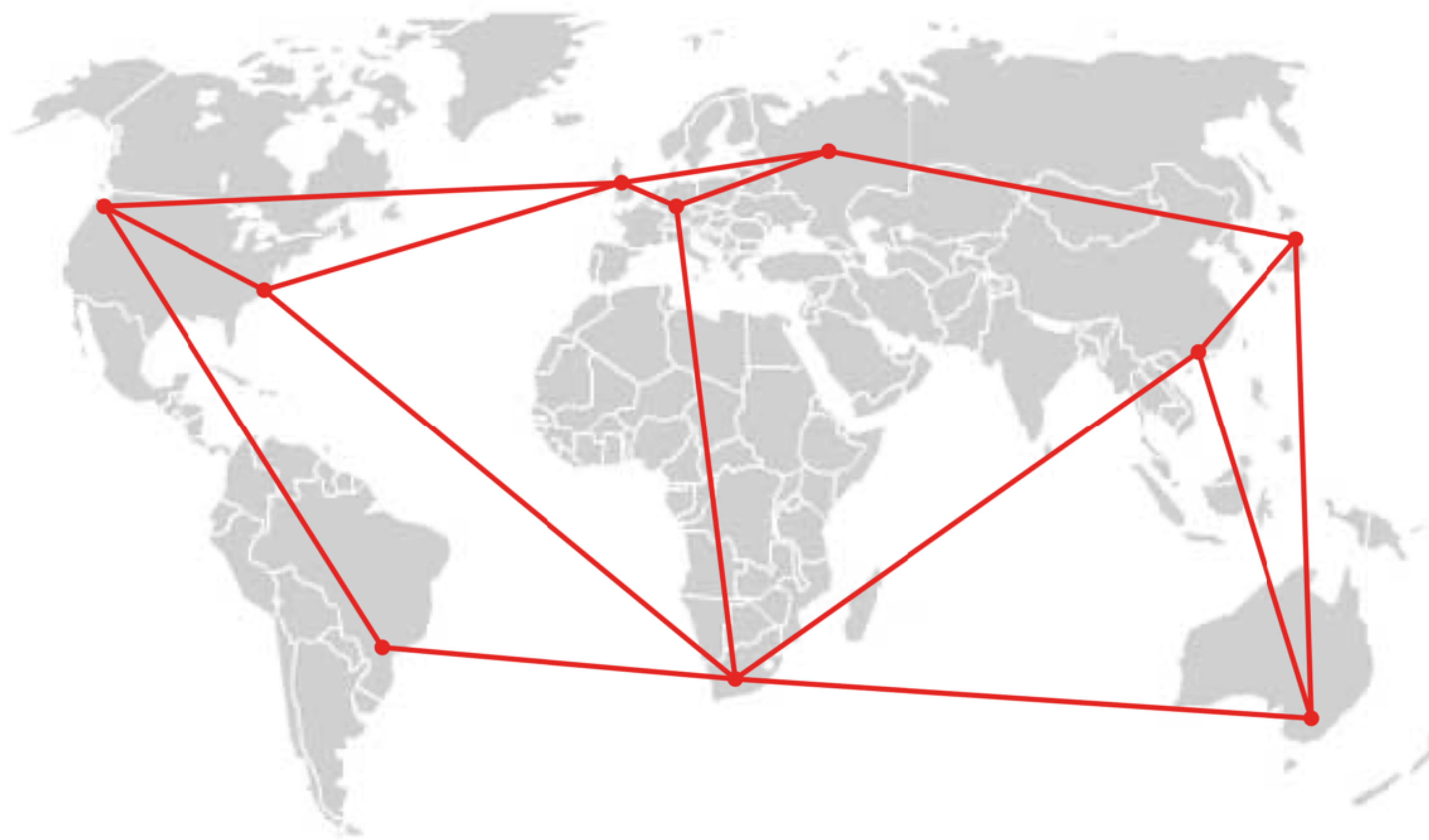
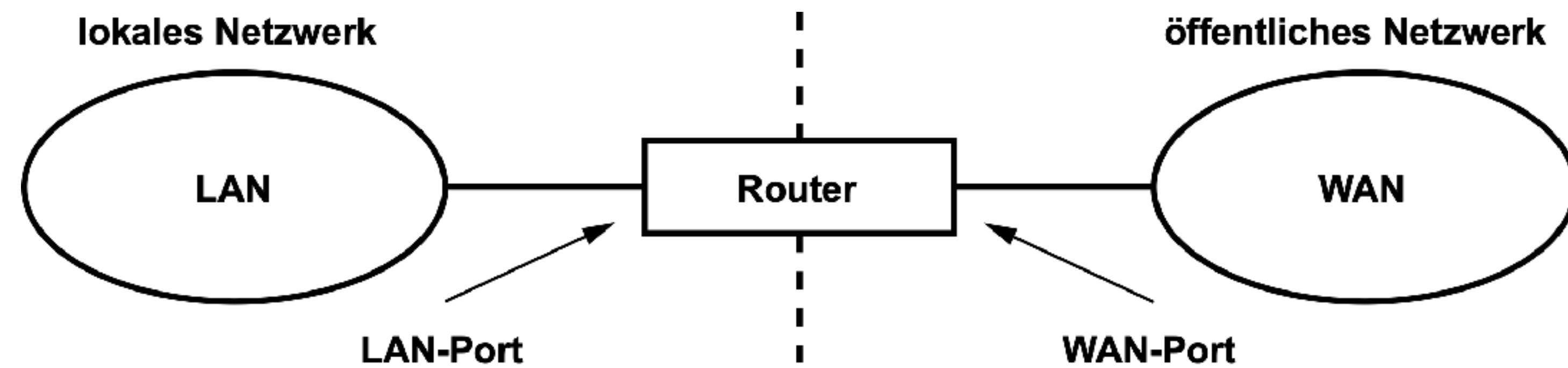
Im ersten Schritt sollte man den Kompatibilitätsmodus in der Konfigurationsoberfläche des WLAN-Routers oder Access-Points identifizieren. Im einfachsten Fall muss man nur einen Haken setzen. Bei einer Fritzbox ist es einfacher, weil explizit genannt ist, welche WLAN-Standards unterstützt werden sollen.

Für eine maximale Kompatibilität sollten die alten WLAN-Standards IEEE 802.11b, 11g und 11n eingeschaltet sein. Es kann aber sein, dass das noch die abschließende Lösung ist.

In einem zweiten Schritt sollte man feststellen, auf welchem Kanal im 2,4-GHz-Frequenzspektrum das WLAN betrieben wird. In den meisten WLAN-Routern wird hier eine automatische Kanalwahl (z. B. Autokanal) eingestellt sein, was in der Praxis auch sinnvoll ist. Leider kann das bedeuten, dass die automatische Wahl auf Kanal 12 oder 13 fallen kann. Manchen WLAN-Geräte unterstützen diese beiden Kanäle nicht.

Es kann sinnvoll sein, den Funkkanal manuell auf Kanal 4 oder 9 einzustellen. Mit diesen beiden Kanälen funktioniert ein Raspberry Pi Pico W auf alle Fälle.

IPv4-Grundlagen



IPv4 ist die Abkürzung für Internet Protocol Version 4. IP kümmert sich unter anderen um die Adressierung und das Routing (Wegfindung) von Datenpaketen innerhalb eines dezentralen Netzwerks. Das Internet ist ein dezentrales Netzwerk.

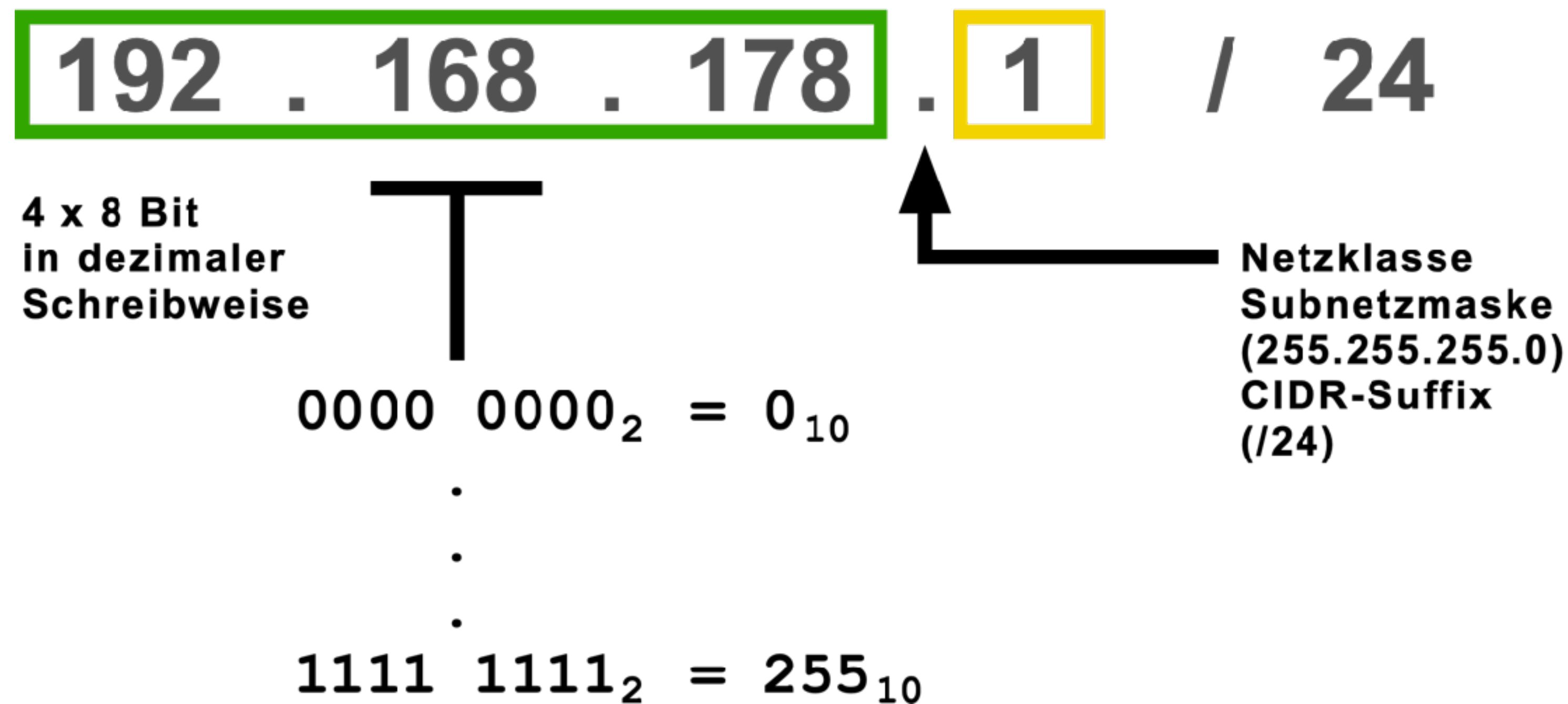
IP braucht man deshalb, weil die WLAN-Technik nur die Kommunikation im selben WLAN oder lokalen Netzwerk ermöglicht. Will man im Internet eine Gegenstelle erreichen, dann braucht man IP. Es ermöglicht die Kommunikation zwischen zwei Hosts über Netzgrenzen hinweg.

Damit Datenpakete ihr Ziel erreichen, müssen sie mit einer IPv4-Adresse adressiert werden. Innerhalb des Netzwerks kümmern sich dann Router darum, dass Datenpakete zum Zielnetzwerk weitergeleitet werden. Dort wird das Paket beim Empfänger zugestellt.

Neben IPv4 gibt es auch den Nachfolger IPv6, der beim Raspberry Pi Pico keine Rolle spielt. IPv6 wird nicht unterstützt.

IPv4-Adresse

Netz-Adresse **Host-Adresse**
Netz-ID **Host-ID**



Um an einem IPv4-Netzwerk teilnehmen zu können, benötigt jeder Host eine IPv4-Konfiguration. Teil der IPv4-Konfiguration ist ein eindeutige IPv4-Adresse, die eine Voraussetzung für die Teilnahme an einem IP-Netzwerk ist. Jedes Datenpaket wird mit der IPv4-Adresse von Absender und Empfänger adressiert.

Eine IPv4-Adresse ist insgesamt 32 Bit lang, wobei jeweils 8 Bit als Dezimalzahl dargestellt wird. Die 4 Dezimalzahlen werden durch einen Punkt voneinander getrennt.

Weil IPv4-Adressen knapp sind, kann nicht jeder Host eine eigene öffentlich erreichbare IPv4-Adresse haben. Private Internet-Anschlüsse haben typischerweise nur eine IPv4-Adresse, die dem Internet-Zugangsrouten zugeteilt ist. Damit alle Hosts in einem Netzwerk eine IPv4-Adresse bekommen können, werden in lokalen Netzwerken oft private IPv4-Adressen verteilt. Der Nachteil davon ist, dass damit keine Kommunikation im Internet möglich ist. Private IPv4-Adressen sind nicht routbar. Damit trotzdem eine Kommunikation im Internet möglich ist, leiht der Internet-Zugangsrouten seine öffentliche IPv4-Adresse für die ausgehenden Verbindungen.

IPv4-Konfiguration

1. IPv4-Adresse, einmalig im Netzwerk
2. Subnetzmaske
3. IPv4-Adresse des Standard-Gateways
(Internet-Zugangsrouter)
4. IPv4-Adresse des DNS-Servers
für die Namensauflösung

Damit ein Host in einem IP-Netzwerk teilnehmen kann muss er konfiguriert werden. Die IPv4-Konfiguration kann auf unterschiedliche Weise erfolgen:

- manuell (statische/feste IPv4-Konfiguration)
- DHCP (halbautomatische IPv4-Konfiguration)

Es gibt noch weitere Möglichkeiten, die beim Raspberry Pi Pico aber keine Rolle spielen.

Wenn man möchte, dass der Raspberry Pi Pico W immer unter der gleichen Adresse erreichbar ist, dann muss die IPv4-Konfiguration manuell erfolgen. Es wäre auch denkbar die IPv4-Adresse für den Pico im DHCP-Server bzw. Internet-Zugangsrouter zu reservieren.

E-Mail-Grundlagen

E-Mail ist die Abkürzung für Electronic Mail, was "Elektronische Post" oder "Elektronischer Brief" bedeutet. Bei einer E-Mail erfolgt das Erstellen, Versenden und Darstellen ausschließlich in elektronischer bzw. digitaler Form. E-Mail ist neben dem Telefon und Messaging ein zentrales Kommunikationsmittel in unserer Gesellschaft. Nachrichten, Diskussionen und Austausch von Dokumenten, das alles ist mit E-Mail möglich.

Die E-Mail-Kommunikation erfolgt über einen Client und einen Server. Hier unterscheidet man zwischen dem Posteingangsserver und dem Postausgangsserver, wobei es sich um unterschiedliche Server handeln kann. Der Ablauf der Kommunikation zwischen einem Mail-Client und einem Mail-Server erfolgt nach dem Client-Server-Prinzip.

Die Unterscheidung zwischen Posteingang und Postausgang liegt daran, weil man bei E-Mail davon ausgehen muss, dass der Nutzer nicht ständig online ist. Beispielsweise, weil der PC ausgeschaltet ist oder weil der empfangende Client nur zeitweise eine Verbindung zum Internet hat.

Die Kommunikation, das Senden und Empfangen von E-Mails, erfolgt über die Protokolle SMTP, POP und IMAP.

- Das SMTP-Protokoll ist für die Übertragung von E-Mails vom SMTP-Client des Absenders zum SMTP-Server (Postausgang) und von dort wiederum zum SMTP-Server des Empfängers zuständig.
- Um die E-Mails abzuholen kontaktiert der Empfänger seinen Posteingangsserver mit POP.
- IMAP hat vom Prinzip die selbe Aufgabe wie POP. Es bietet jedoch mehrere Vorteile. Zum Beispiel ermöglicht IMAP das Verwalten von E-Mails auf dem Mail-Server.

E-Mail-Konfiguration

- Benutzername
- Passwort
- Hostname Postausgang
- Port Postausgang
- Hostname Posteingang
- Port Posteingang

Hinweis: Wenn man keine E-Mails empfangen möchte, dann werden diese Zugangsdaten auch nicht benötigt.

Zum Senden und Empfangen von E-Mails braucht man einen E-Mail-Client oder einen Programmcode, der für die Kommunikation mit dem Posteingang und Postausgang jeweils eine eigene Konfiguration braucht.

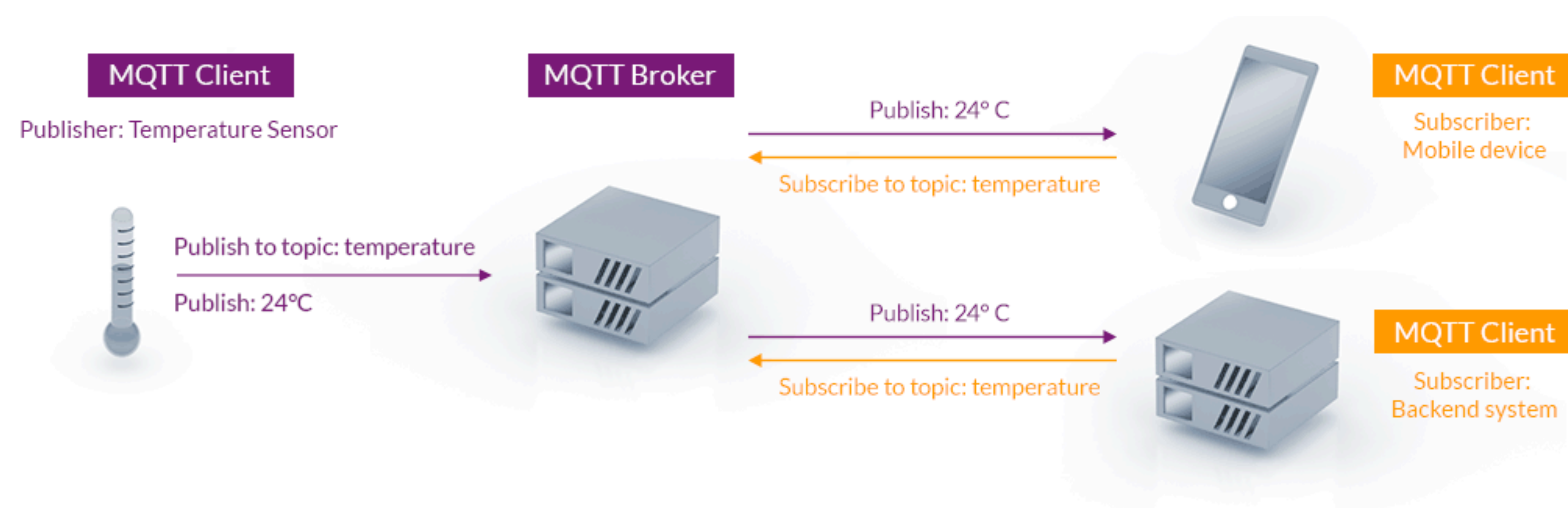
- Zum Senden von E-Mails braucht der E-Mail-Client oder der Programmcode, der eine E-Mail senden soll, die Zugangsdaten (Benutzername und Passwort) und Adresse des Postausgang-Servers.
- Zum Empfang von E-Mails braucht der E-Mail-Client oder der Programmcode, der eine E-Mail empfangen soll, die Zugangsdaten (Benutzername und Passwort) und Adresse des Posteingang-Servers.

Die Konfiguration muss man beim jeweiligen Mail-Provider anfragen bzw. kann diese Informationen über dessen Web-Konfigurationsoberfläche einsehen.

MQTT-Grundlagen



Message Queue Telemetry Transport, kurz MQTT, ist ein äußerst einfach aufgebautes Kommunikationsprotokoll für den Nachrichtenaustausch zwischen Geräten in Umgebungen mit geringer Bandbreite und instabilen Verbindungen. Typischerweise lassen sich mit MQTT Sensordaten, wie Temperatur oder Füllstände, übertragen. Aber auch klassische Nachrichten oder Kurzmitteilungen. Es eignet sich demnach besonders für Internet of Things (IoT) und Smart Home.



Die MQTT-Architektur kennt 3 Rollen. Eine Rolle ist die des Publishers, der Daten senden will. Eine weitere Rolle ist die des Subscribers, der Daten empfangen will. Zwischen Publisher und Subscriber befindet sich der Broker, der Daten von Publishern entgegennimmt und an die Subscriber verteilt.

MQTT-Kommunikation

Wie finden Publisher und Subscriber mit MQTT zueinander?

Ein Publisher sendet seine Nachrichten dem Broker mit einem bestimmten Topic (Name/Thema/Bezeichner). Die Subscriber, die Nachrichten zu einem bestimmten Topic haben wollen, melden sich beim Broker an und abonnieren das Topic. Der Broker kümmert sich dann darum, dass die Subscriber die Nachrichten von den abonnierten Topics bekommen. Durch Quality-of-Service-Level kann eine Zustellung auch garantiert werden.

MQTT-Sicherheit

Eine sichere Kommunikation mit MQTT erfordert die Authentifizierung der Gegenstellen und die Verschlüsselung der Datenpakete. Beides ist mit MQTT möglich, erfordert jedoch die entsprechende Konfiguration und Unterstützung der Software und im implementierten Programmcode.

Ohne Verschlüsselung erfolgt die Kommunikation im Klartext und kann abgehört und manipuliert werden. Ohne Authentifizierung ist unklar von wem die Daten wirklich kommen und somit wären Steuerungen manipulierbar.

MQTT-Implementierung in MicroPython

Damit ein Raspberry Pi Pico W per MQTT kommunizieren kann, muss im Programmcode die Funktionsweise von MQTT implementiert werden. Es macht allerdings nur wenig Sinn, wenn jeder Anwender seine eigene Implementierung schreibt. Es gibt bereits fertige und geprüfte Implementierungen. Diese MicroPython-Bibliotheken sind frei verfügbar und lassen sich über den Paket-Manager in Thonny installieren oder von Github herunterladen und auf dem Raspberry Pi Pico speichern.

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-Highlighting 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: Beliebiger Computer mit Zubehör (Maus, Tastatur, Bildschirm)
- Elektronik: Raspberry Pi Pico mit verschiedenen Bauteilen, Steckbrett, Verbindungskabel und Micro-USB-Kabel
- Programmiersprache: MicroPython (muss zuerst auf dem Pico installiert werden)
- Editor: Thonny Python IDE (muss auf dem Computer installiert sein)
- Internet: von Vorteil, aber nicht zwingend erforderlich

Raspberry Pi Pico programmieren

Editor: Thonny Python IDE

Das Schreiben und Programmieren eines Programms erfolgt in einem Editor. Empfohlen wird die Thonny Python IDE.

Diesen Editor gibt es für verschiedene Betriebssysteme. Wenn Du mit der Desktop-Version von Raspberry Pi OS arbeitest, dann erreichst Du das Programm im Menü über das Untermenü „Entwicklung“. In anderen Betriebssystemen muss Thonny erst noch installiert werden.

Stelle zuerst sicher, dass die Thonny Python IDE installiert ist. Start das Programm testweise und beende es anschließend wieder.

<https://thonny.org/>

Programmiersprache: MicroPython

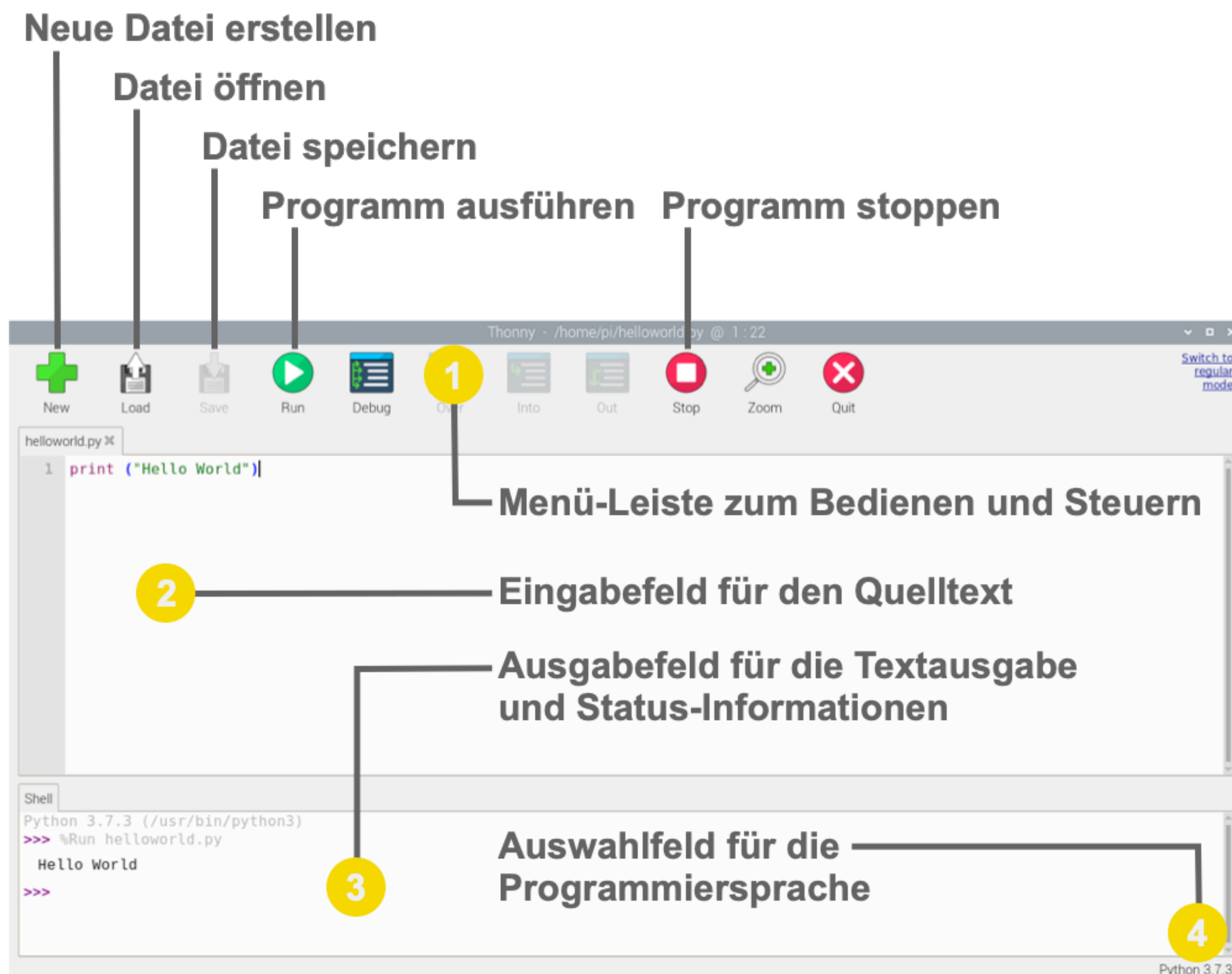
Standardmäßig ist der Raspberry Pi Pico für die Programmierung mit C und C++ eingerichtet.

Für Anfänger ist MicroPython zu empfehlen. Das ist eine einfachere Version der Programmiersprache Python, die speziell für Mikrocontroller entwickelt wurde.

Wenn der Pico mit MicroPython programmiert werden soll, dann muss MicroPython zuerst auf dem Board installiert werden. Der Vorgang ist einmalig und besteht nur darin, eine Datei aus dem Internet herunterzuladen und auf den Pico zu kopieren.

<https://docs.micropython.org/en/latest/rp2/quickref.html>

Bedienen der Thonny Python IDE



Die Thonny Python IDE verfügt über alle erforderlichen Funktionen und Bedienelemente, um effektiv mit dem Pico und MicroPython 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-Infos

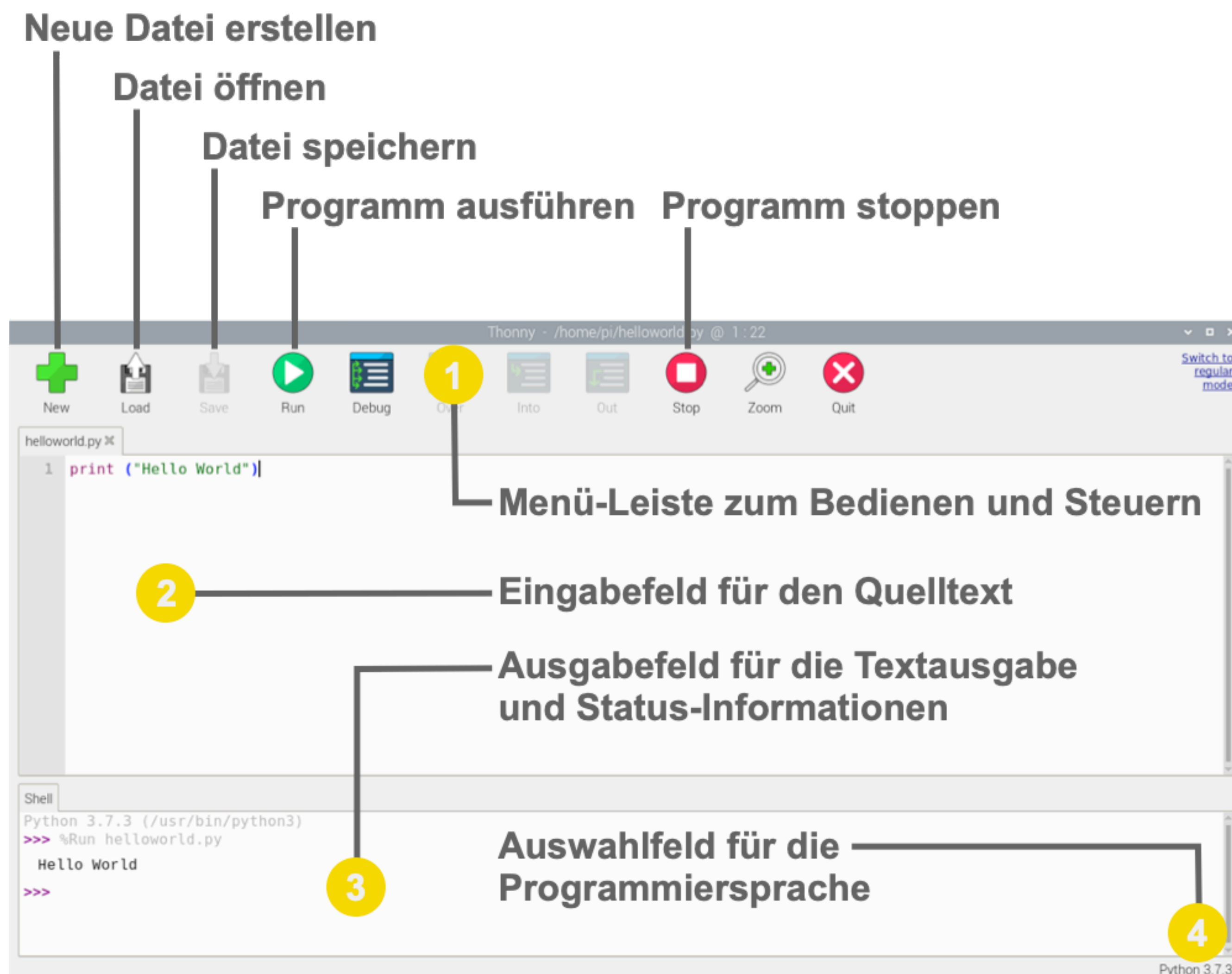
Wichtige Funktionen der Menü-Leiste

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

Auswahl der Programmiersprache

Rechts unten befindet sich die Anzeige der aktuell ausgewählten Programmiersprache. Mit einem Klick auf diese Status-Anzeige öffnet sich ein Menü mit weiteren Programmiersprachen und Python-Versionen. Für die Programmierung des Raspberry Pi Pico ist hier „MicroPython“ auszuwählen.

MicroPython auf dem Pico installieren (1)



Verbinde Deinen Pico mit Deinem Computer

Hilfreich ist es, wenn Du Deinen Pico immer zuerst über das Micro-USB-Kabel mit Deinem Computer verbindest und erst danach die Thonny Python IDE startest.

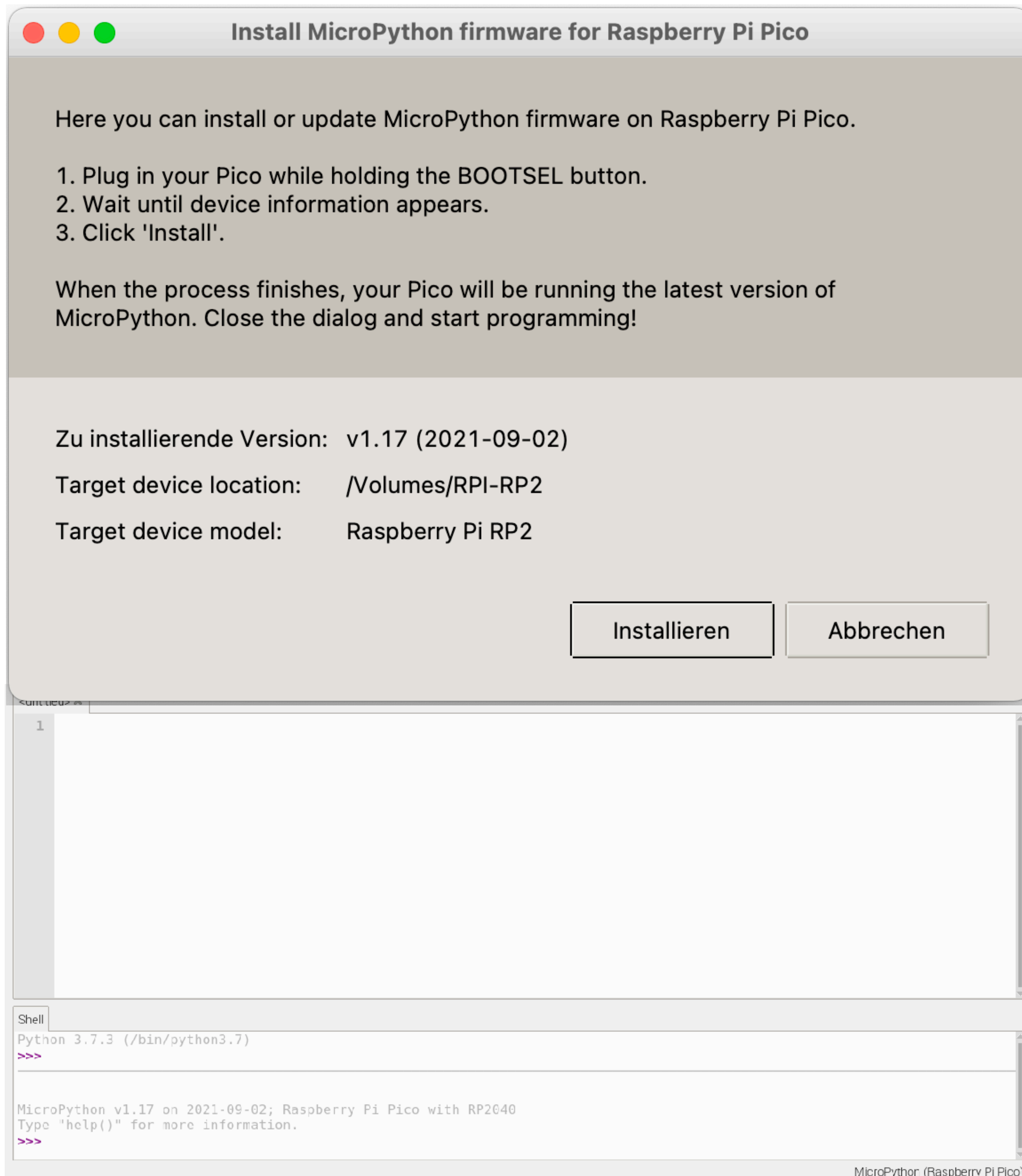
Es kann sein, dass Du keine Rückmeldung beim Einstecken des Picos von Deinem Betriebssystem bekommst. Das ist in Ordnung.

Thonny-Editor starten und MicroPython auswählen

Wenn der Pico mit Deinem Computer verbunden ist, kannst Du den Thonny-Editor starten. Diese Reihenfolge hat den Vorteil, dass der Thonny-Editor den Pico automatisch erkennt. Hier ist zu beachten, dass die Thonny Python IDE normalerweise auf Python eingestellt ist. Um den Pico mit MicroPython programmieren zu können, muss in der Thonny Python IDE noch die richtige Programmiersprache und Version ausgewählt werden.

Klicke dazu rechts unten auf die Bezeichnung „Python“ oder ähnlich und wähle dort „MicroPython“ für Raspberry Pi Pico aus. Danach ist das Programm umgestellt und der Pico wird dabei automatisch erkannt, wenn er per USB mit Deinem Computer verbunden ist.

MicroPython auf dem Pico installieren (2)



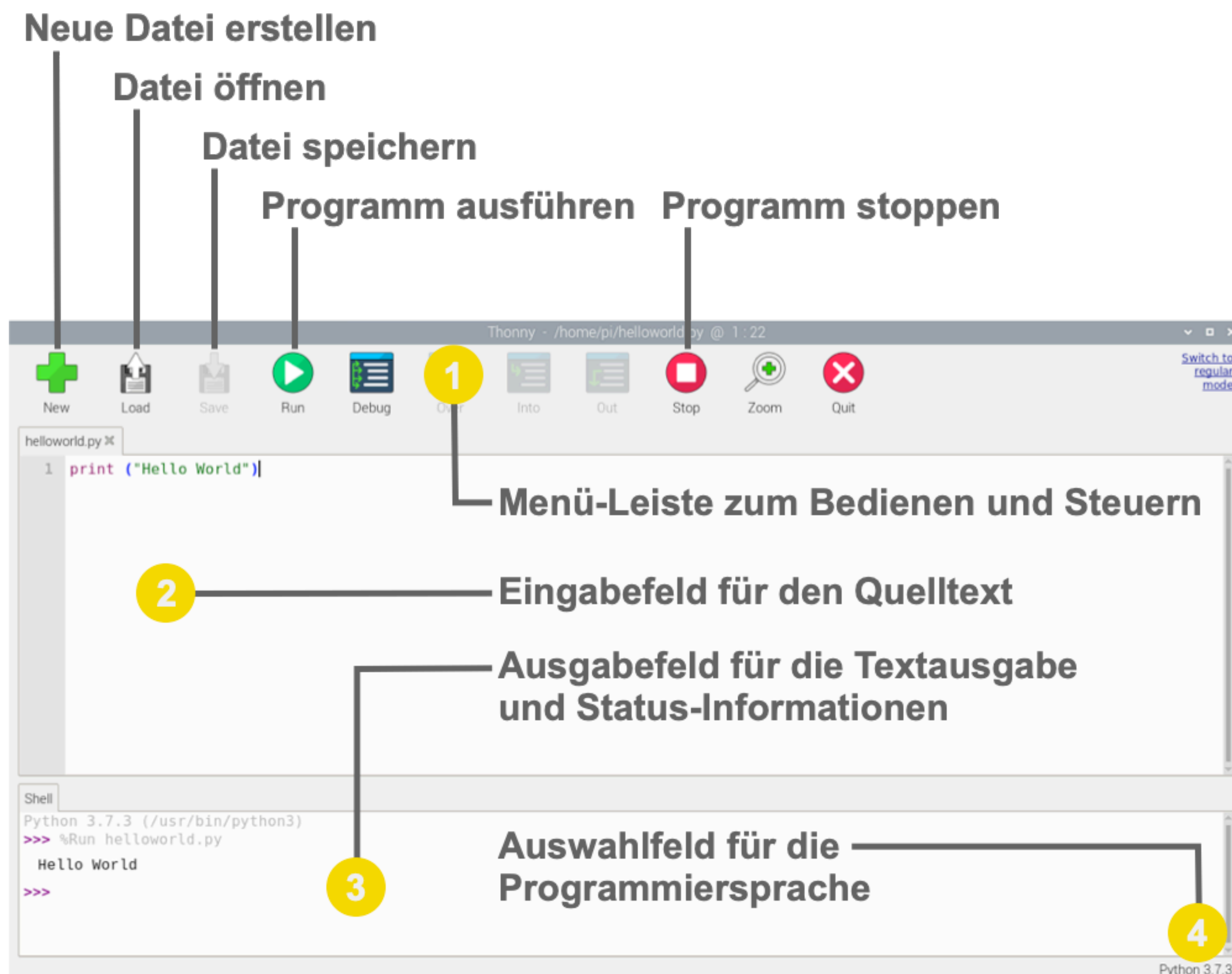
MicroPython auf dem Raspberry Pi Pico installieren

Beim erstmaligen Erkennen eines Raspberry Pi Picos wird die Thonny Python IDE eine Firmware aus dem Internet nachladen wollen. Das kannst Du durch Klicken auf „Install“ bestätigen. Der Vorgang ist in der Regel innerhalb weniger Sekunden erledigt. Danach ist Dein Raspberry Pi Pico zum Programmieren mit MicroPython und dem Thonny-Editor bereit.

Wichtig

Der Thonny-Editor ist mit dem Pico dann zum Programmieren bereit, wenn im Feld „Shell“ bzw. „Kommandozeile“ eine Meldung mit „MicroPython“ und „Raspberry Pi Pico with RP2040“ oder „Raspberry Pi Pico W with RP2040“ erscheint. Wenn nicht, dann gibt es dafür eine einfache Lösung: Wenn offensichtlich keine Verbindung besteht, dann kannst Du durch Klicken auf „STOP“ in der Menü-Leiste die Verbindung manuell herstellen.

Programmieren mit der Thonny Python IDE



Neue Datei erstellen

Am Anfang wirst Du ein neues Programm immer mit einer neuen Datei beginnen indem Du auf „New“ klickst (grünes Plus). Anschließend kannst Du einen fertigen Quelltext in das Textfeld kopieren oder Dein Programm von Grund auf neu schreiben.

Programm speichern

Zum Speichern Deines Programms dient im Thonny-Editor die „Save“-Funktion (Diskette mit Pfeil nach unten). Nachdem Du darauf geklickt hast, hast Du die Möglichkeit das Programm entweder auf Deinem Computer oder auf dem Pico zu speichern. Zum Ausführen auf dem Pico musst Du Dein Programm auf dem Pico speichern. Wichtig ist, dass Du die Datei mit der Dateiendung „.py“ speicherst.

Wichtig: Der Thonny-Editor merkt sich beim Öffnen und Speichern von Dateien den Speicherort.

Programm ausführen und stoppen

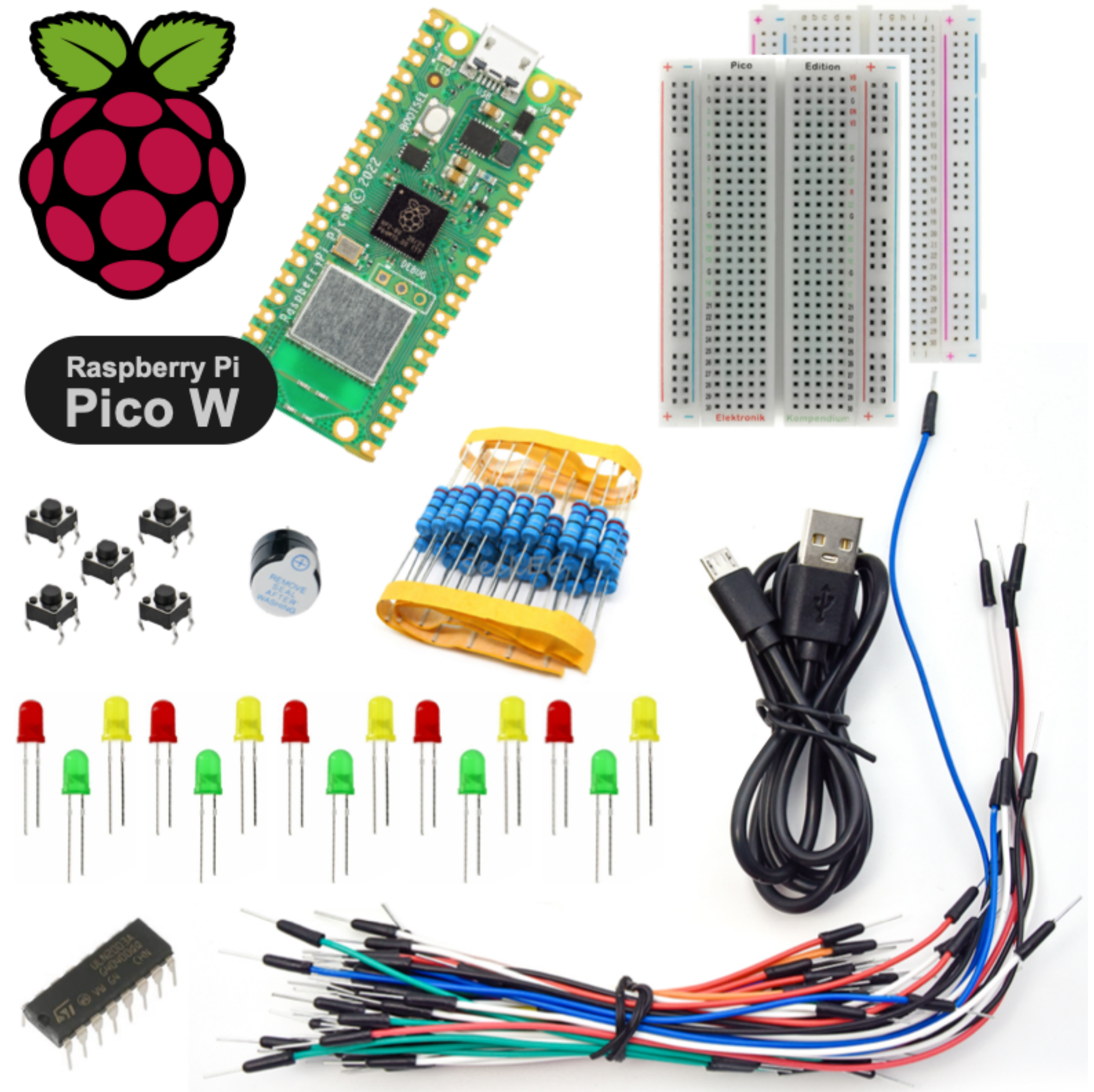
Das Programm startet dann, wenn Du auf „Run“ klickst (grüner Kreis mit weißem Dreieck). Wenn Du das Programm beenden willst, klickst Du auf „Stop“ (roter Kreis mit weißem Rechteck).



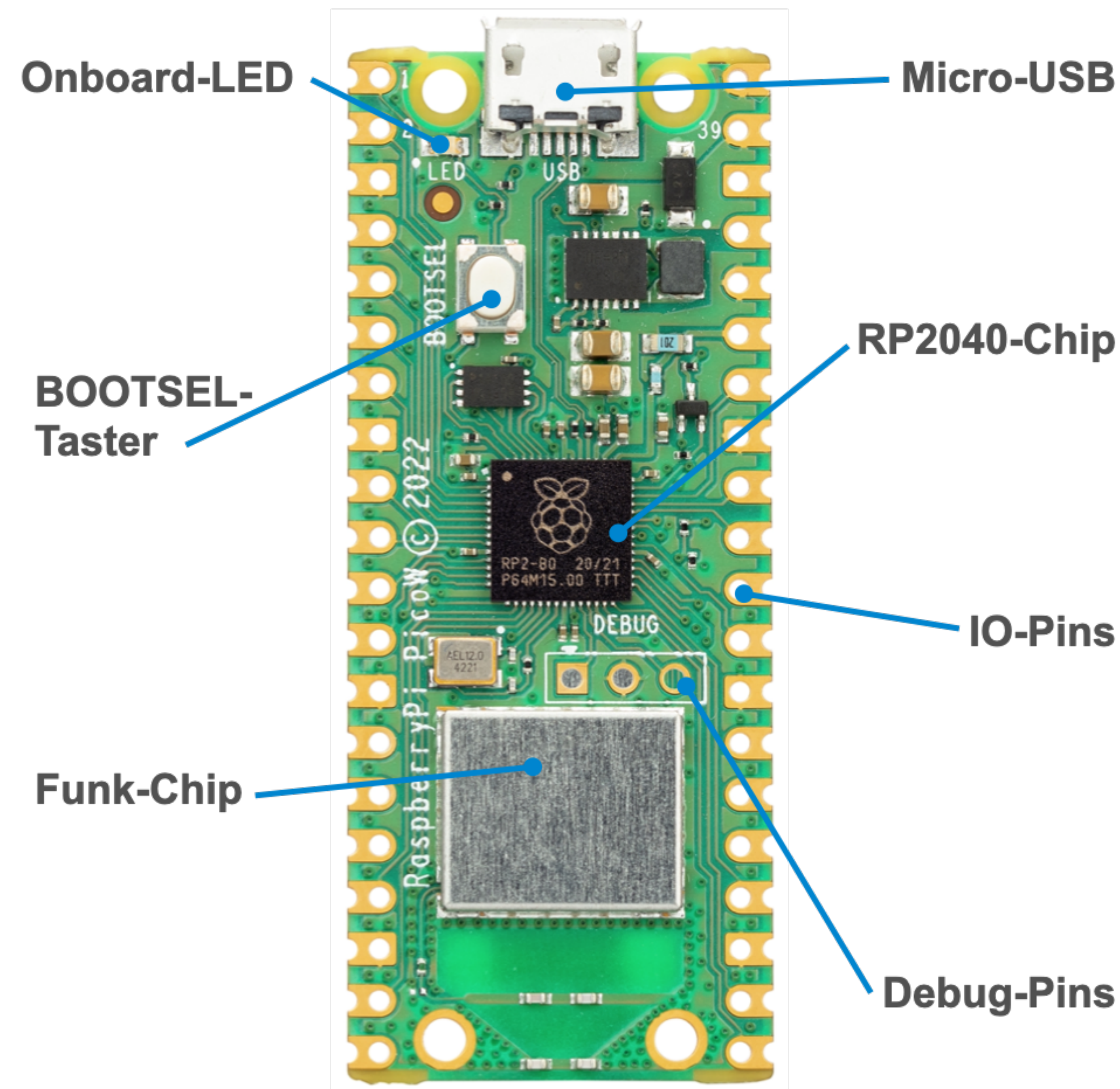
Bauteile

Bauteile im Elektronik-Set Pico WLAN Edition

- Raspberry Pi Pico W (Mikrocontroller-Modul)
- USB-Kabel
- Steckbrett „Pico Edition“
- Verbindungskabel
- Taster
- Widerstände
- Leuchtdioden
- Potentiometer
- (Summer)
- ULN2003A (IC) für zukünftige Anwendungen



Raspberry Pi Pico W (Modul)



Der Pico wird über ein USB-Kabel mit einem Computer verbunden. Der Pico wird dabei mit Strom versorgt und meldet sich dann am Computer an. Die Programmierung erfolgt über eine virtuelle serielle Schnittstelle.

- programmierbarer Mikrocontroller
- Modul mit Chip, Speicher, WLAN und Anschlüssen

Der Raspberry Pi Pico W ist eine Platine mit einem Mikrocontroller drauf, zusätzlichen Bausteinen für die externe Beschaltung und Programmierung. Mit zwei 20-poligen Stiftleisten ist die Platine Steckbrett-kompatibel, wodurch die GPIO-Pins für die externe Beschaltung leichter zugänglich sind. Auf einem Steckbrett lassen sich Kabel und andere Bauteile durch Stecken leichter hinzufügen oder entfernen.

An einem Ende der Platine befindet sich ein Micro-USB-Anschluss (Typ B) zur Stromversorgung und zur Programmierung über einen Computer.

Neben dem USB-Anschluss befindet sich eine Onboard-LED, die für die eigene Programmentwicklung genutzt werden kann. Daneben wiederum befindet sich ein BOOTSEL-Taster, der während der Programmierung des Mikrocontrollers benötigt wird.

USB-Kabel



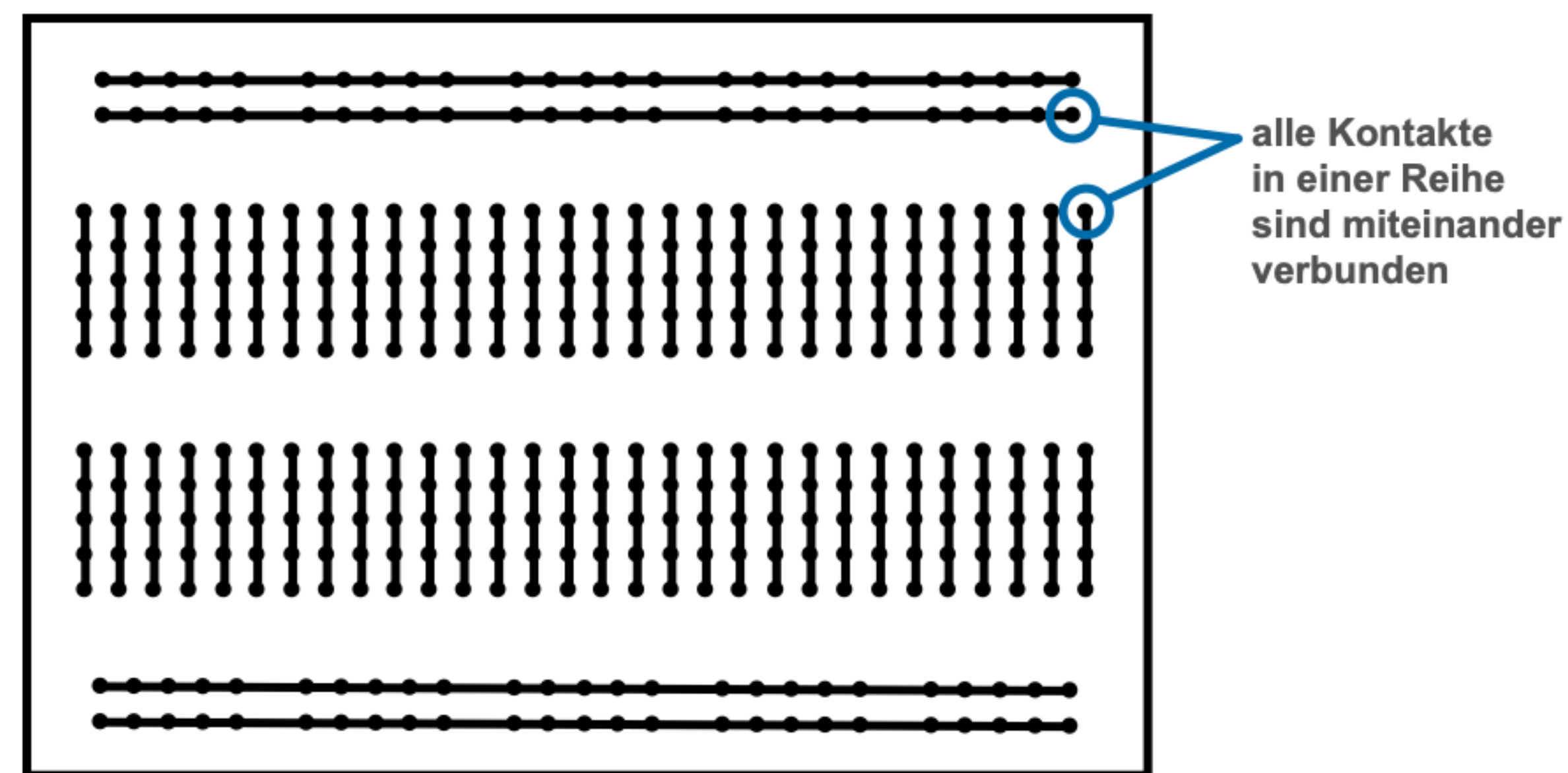
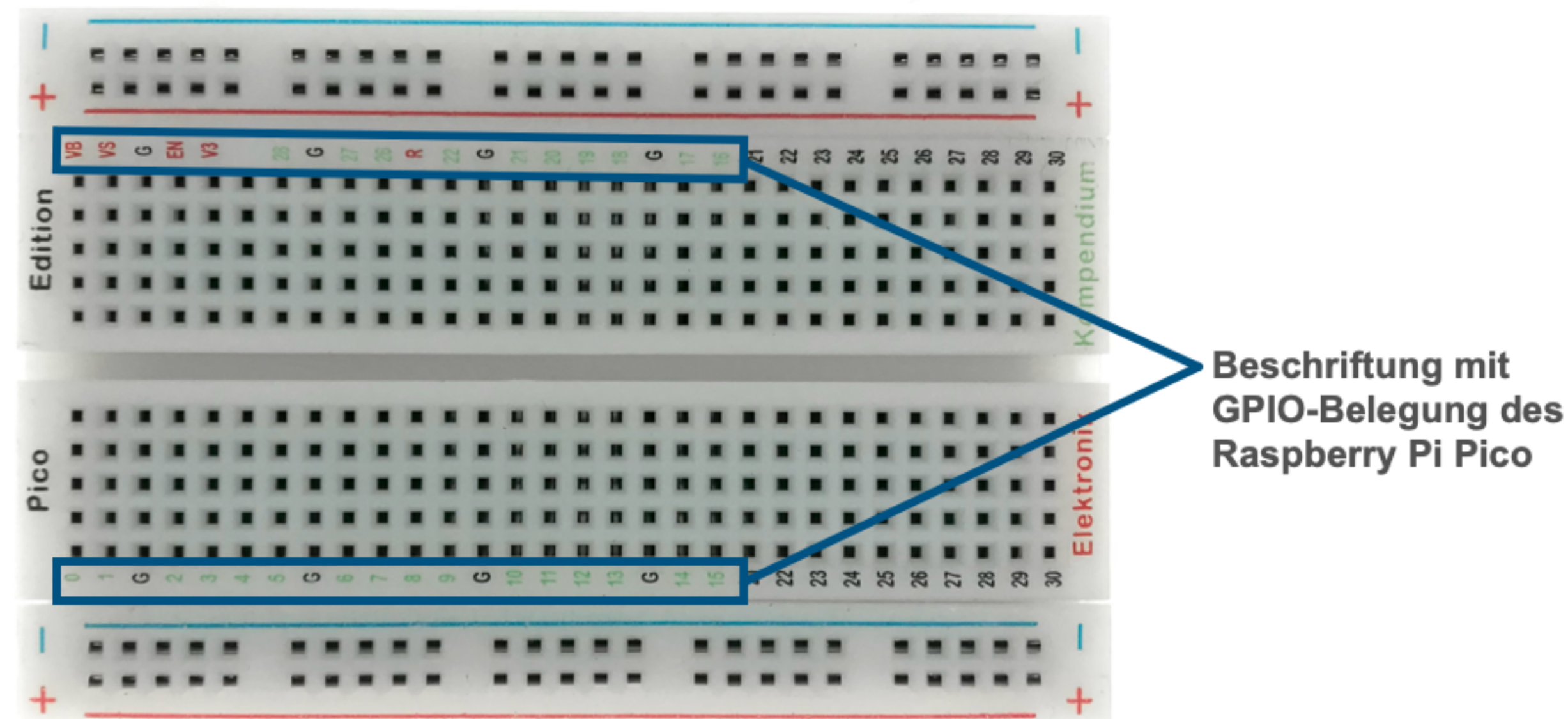
- Verbindungskabel zwischen Computer (Host) und Raspberry Pi Pico (Endgerät)
- Verbindungskabel mit Datenleitungen
- Stromversorgung für den Pico oder ein anderes Endgerät mit 5 Volt

USB steht für Universal Serial Bus. Es ist eine universelle Schnittstelle, um Endgeräte und Peripherie mit einem Computer zu verbinden. Typischerweise Tastatur, Maus, USB-Sticks, Drucker und ähnliches. Eine Besonderheit dieser Schnittstelle ist, dass sie nicht nur Datenleitungen hat, sondern auch die Energieversorgung für das Endgerät übernehmen kann.

An einem Ende der Platine des Pico befindet sich ein Micro-USB-Anschluss (Typ B) zur Stromversorgung und zur Programmierung über einen Computer.

Der Pico wird über den USB-Kabel mit einem Computer verbunden. Der Pico meldet sich dabei als Laufwerk am Computer an und kann durch Speichern einer Datei mit dem Programmcode auf diesem Laufwerk programmiert werden.

Steckbrett für den Raspberry Pi Pico



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

- lötfreies Experimentieren durch Steckverbindungen
- aufgedruckte GPIO-Nummern des Pico

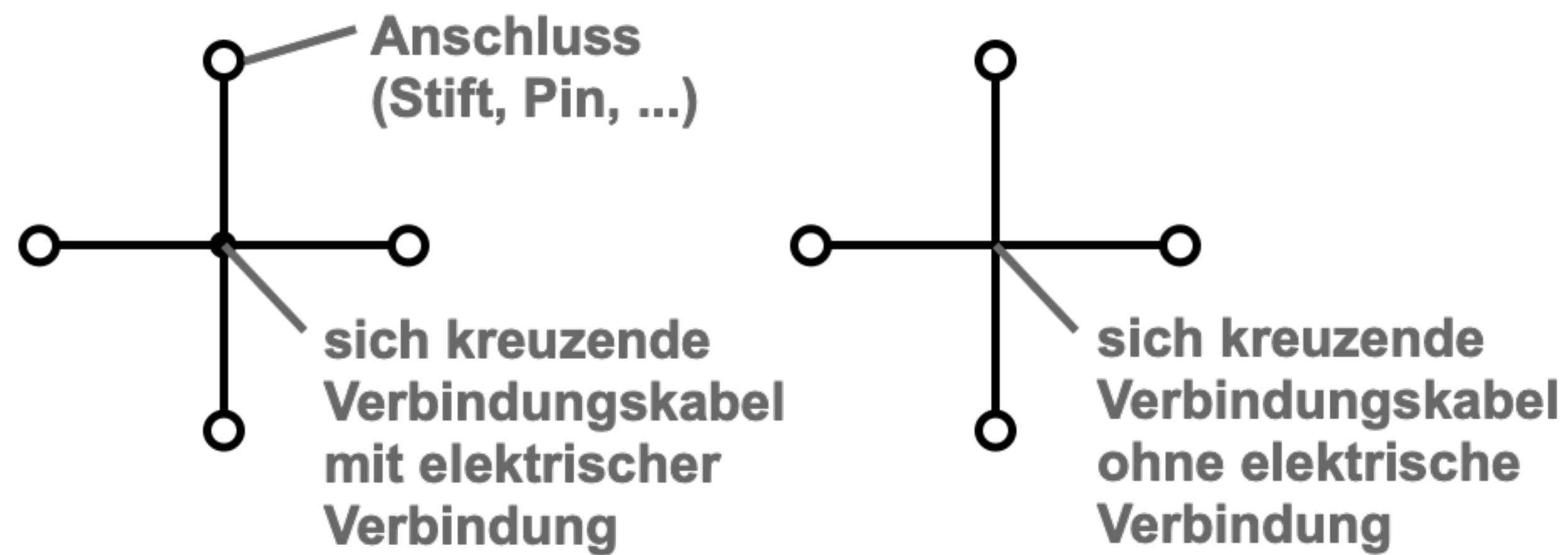
Ein Steckbrett oder Steckboard ist eine Experimentier-Platine, auf der sich elektronische Bauteile durch Stecken mechanisch befestigen und zu Schaltungen elektrisch verbinden lassen.

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. 1/10-Zoll-Raster, die reihenweise miteinander verbunden sind. Über diese Kontaktreihen und zusätzlich steckbare Drahtbrücken werden die einzelnen Bauteile miteinander verbunden.

In der Mitte des Steckbretts befindet sich ein breiter Steg, damit die Anschlüsse von integrierten Schaltungen (ICs) in DIL-Bauform voneinander getrennt sind.

Eine spezielle Beschriftung an den Steckreihen kennzeichnet die GPIO-Nummern des Raspberry Pi Pico.

Steckverbindungskabel

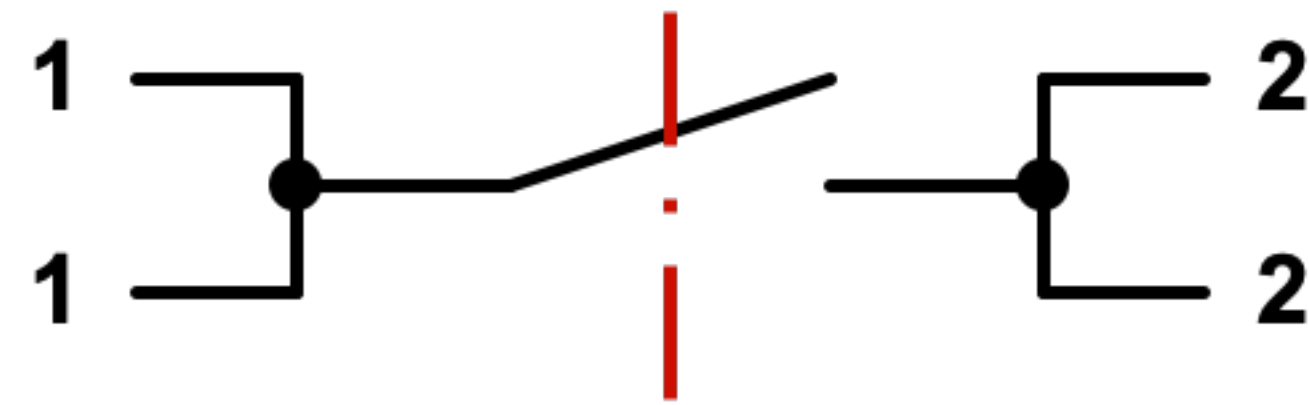


- elektrische Verbindungen zwischen Bauelementen, Schaltungen und Geräten

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

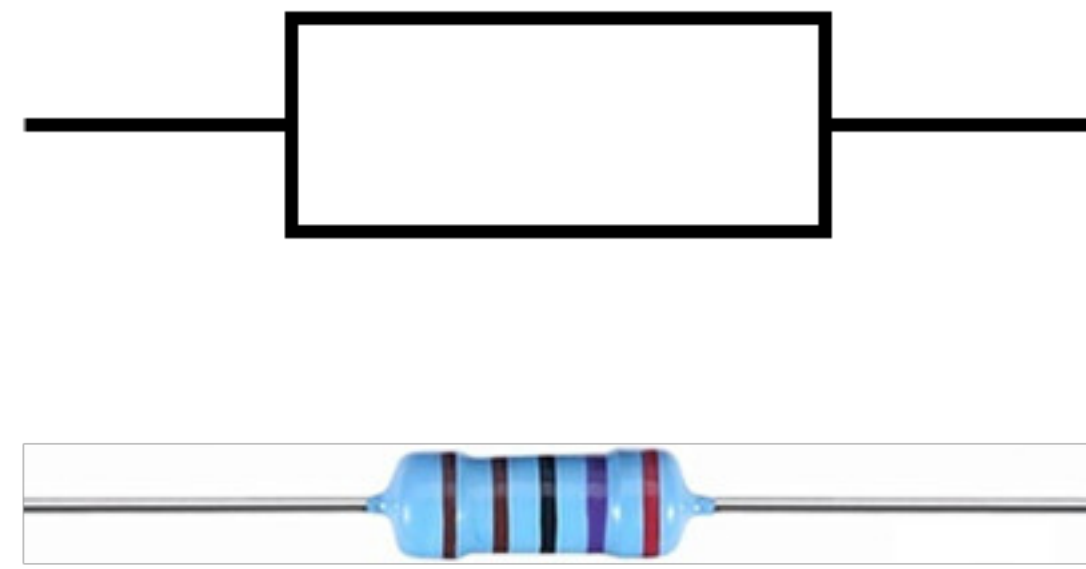
Durch einen Taster kann ein Stromkreis geschlossen und unterbrochen werden. Damit steuert man, ob im Stromkreis ein Strom fließt oder nicht.

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



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.

- 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.

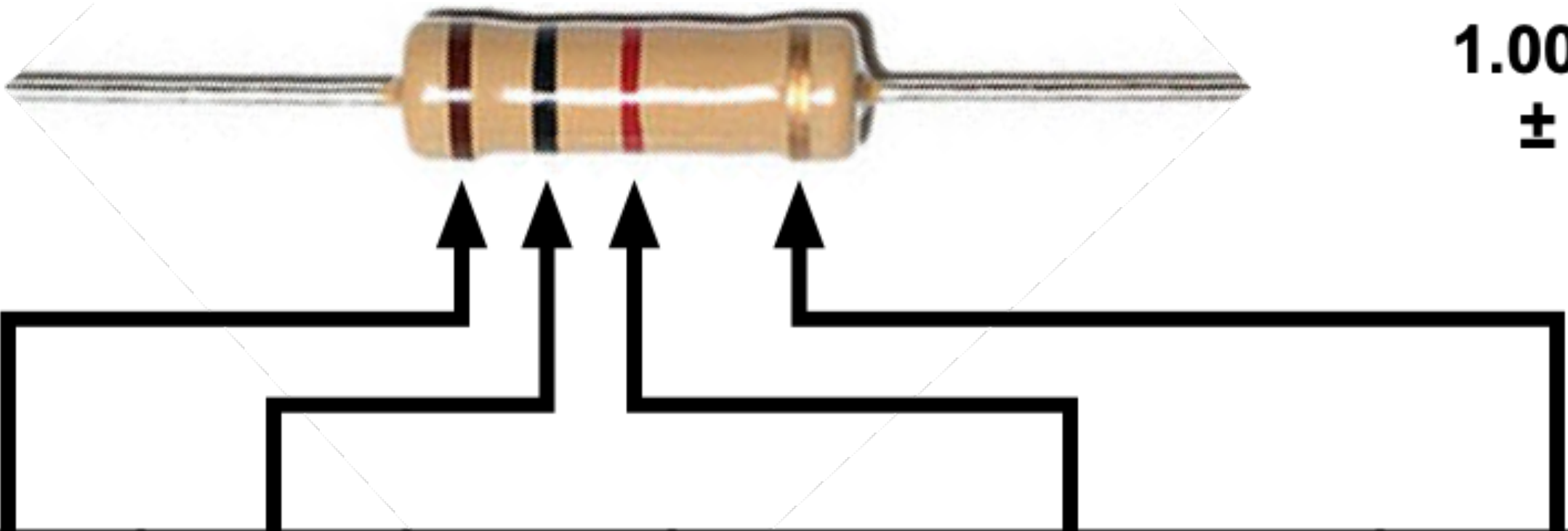
Um einen bestimmten Widerstandswert zu errechnen, verwendet man das Ohmsche Gesetz.

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)


4 Ringe



**1.000 Ω
± 5 %**

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 %

5 Ringe



**2.700 Ω
± 1 %**

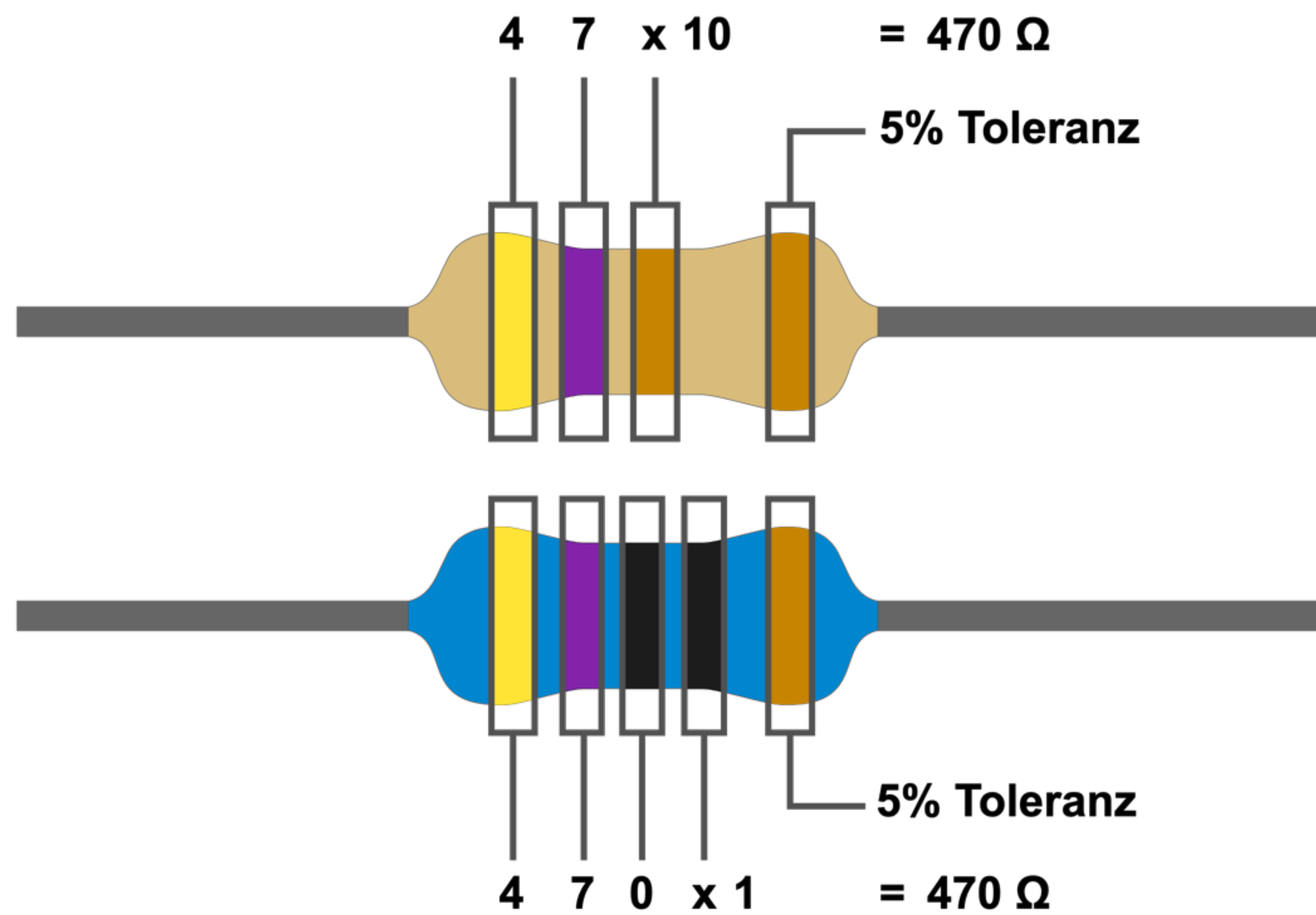
Widerstände werden in der Regel durch Farbringe gekennzeichnet. Je nach Art des Widerstandsmaterials, werden 4 oder 5 Ringe verwendet.

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.

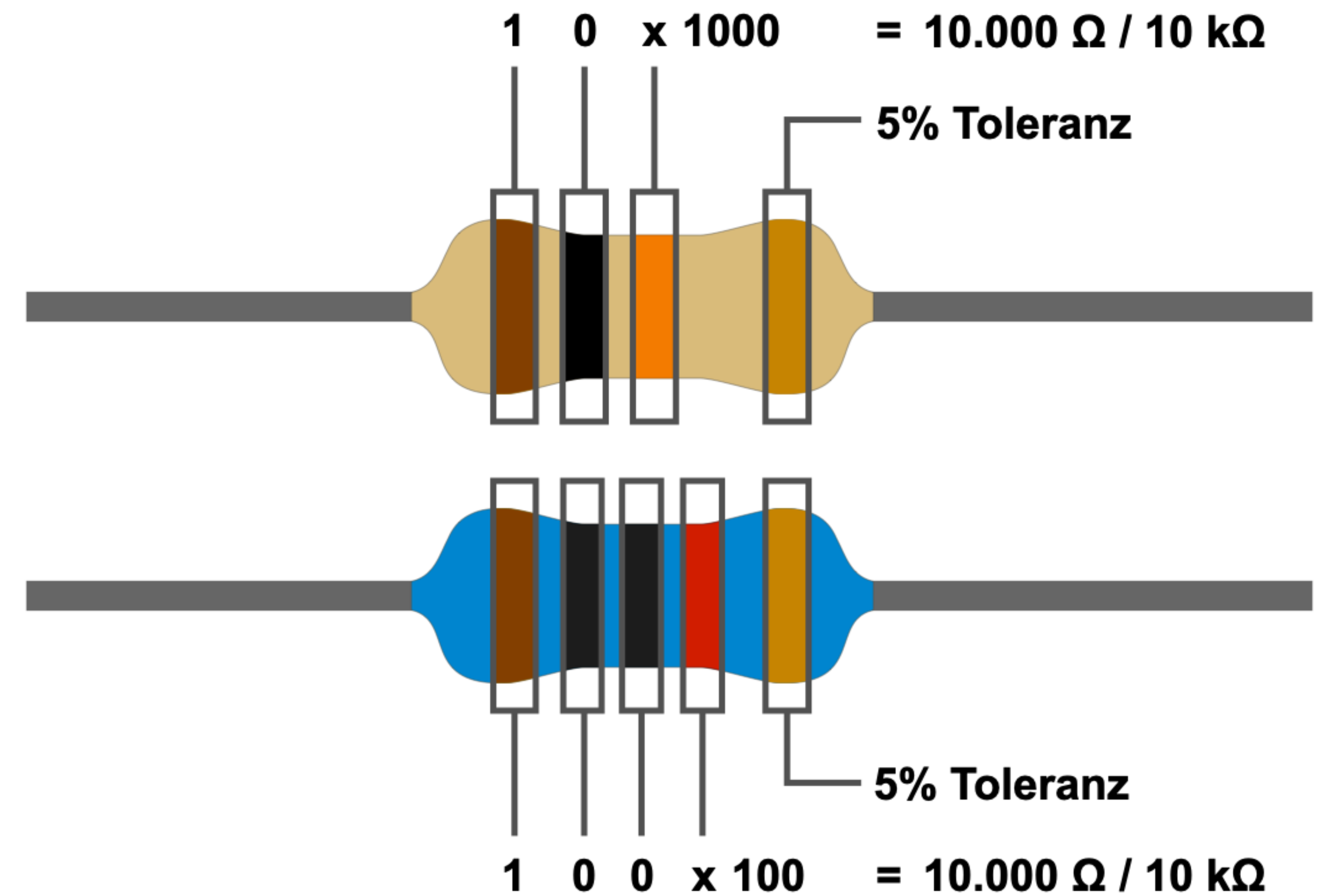
Die Ringe entsprechen einem internationalen Farbcode zur Kennzeichnung und Bestimmung des Widerstandswertes in Ω, sowie der Toleranz dieses Wertes. Um die Werte abzulesen, behilft man sich mit einer Tabelle.

Kennzeichnung von Widerständen (2)

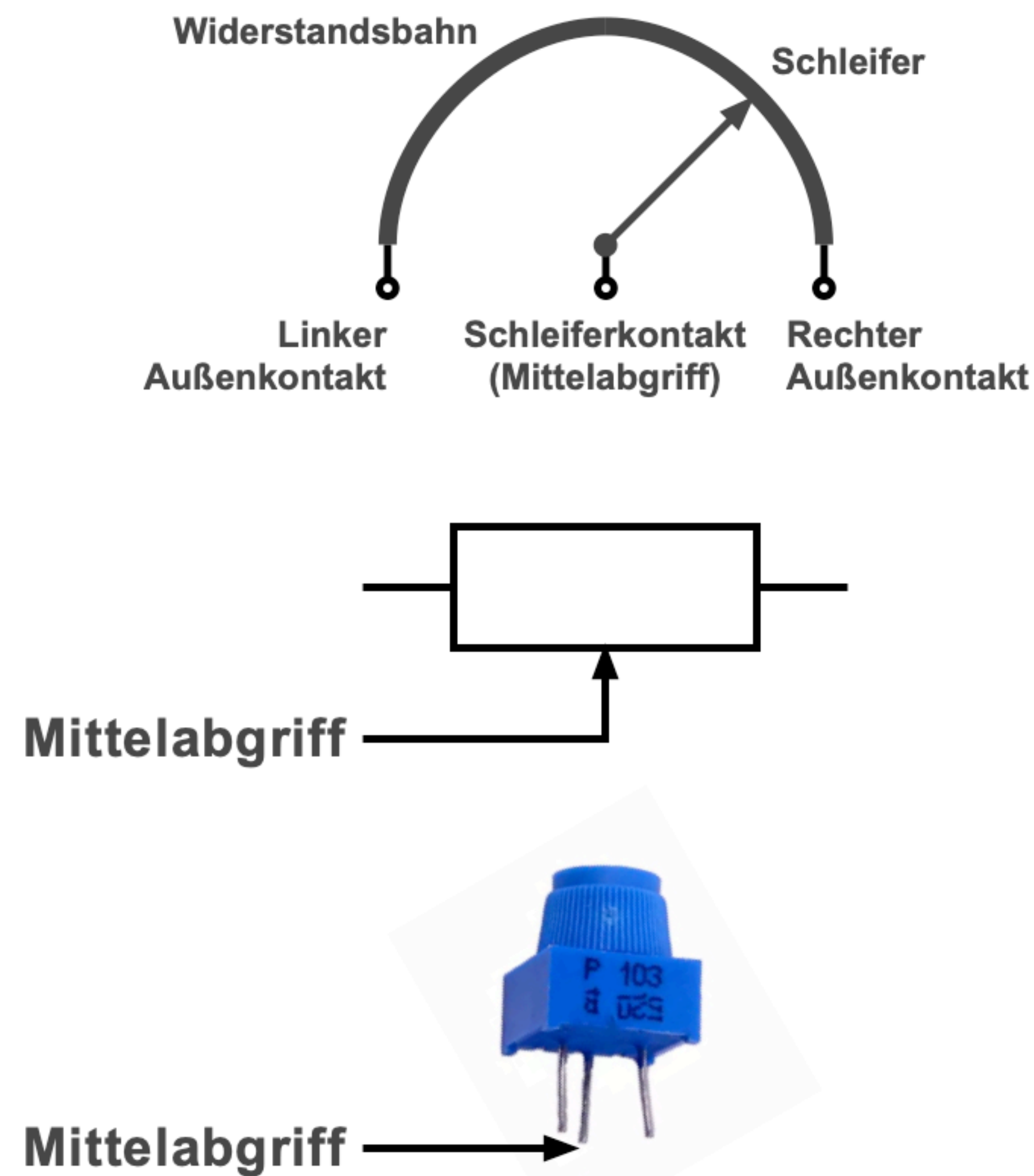
Widerstand mit 470 Ohm



Widerstand mit 10 kOhm



Potentiometer (Poti)



- einstellbarer Widerstand
- Einstellen des Stroms
- Einstellen der Spannung

Ein Potentiometer, kurz Poti, ist ein passives Bauelement dessen Widerstandswert sich stufenlos einstellen lässt. Über den Widerstandskörper wird ein Kontakt (Schleifer) geführt, über dessen Position man einen bestimmten Widerstand einstellen und abgreifen kann. Dazu hat das Potentiometer nicht nur zwei, sondern drei Anschlüsse. Zwei für den Widerstand und den dritten für den Abgriff.

Ein Potentiometer kann man sich wie einen Spannungsteiler aus zwei in Reihe geschalteten Widerständen vorstellen. Mit einem Drehregler stellt man das Verhältnis der beiden Widerstände ein.

Kennzeichnung von Potentiometern (Potis)



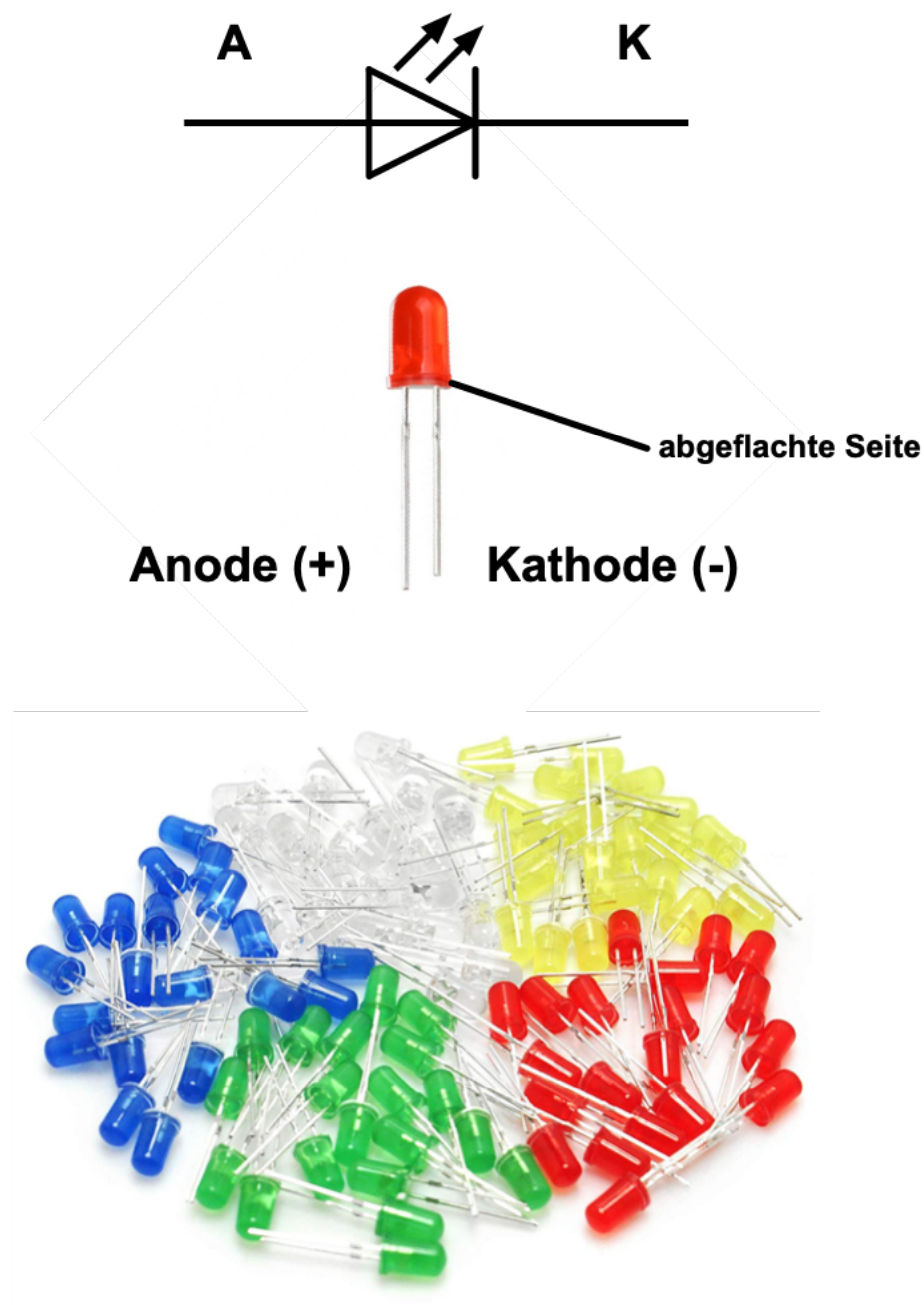
$$\begin{aligned} &= 10.000 \Omega \\ &= 10 \text{ k}\Omega \end{aligned}$$

Manchmal ist auf dem Potentiometer der Widerstandswert aufgedruckt. Manchmal ist die Kennzeichnung kodiert, also eine Kurzform. Dann sind die ersten zwei Ziffern der Wert in Ohm. Eine dritte Ziffer ist der Multiplikator, also die Anzahl der Nullen, die man dem Wert anfügt. Ab drei Nullen rechnet man in Kiloohm (kOhm) um.

Beispiele:

101:	10	x	10	=	100	Ohm	
102:	10	x	100	=	1.000	Ohm	= 1 kOhm
103:	10	x	1.000	=	10.000	Ohm	= 10 kOhm
104:	10	x	10.000	=	100.000	Ohm	= 100 kOhm
105:	10	x	100.000	=	1.000.000	Ohm	= 1 MOhm
151:	15	x	10	=	150	Ohm	
152:	15	x	100	=	1.500	Ohm	= 1,5 kOhm
153:	15	x	1.000	=	15.000	Ohm	= 15 kOhm
154:	15	x	10.000	=	150.000	Ohm	= 150 kOhm
201:	20	x	10	=	200	Ohm	
202:	20	x	100	=	2.000	Ohm	= 2 kOhm
203:	20	x	1.000	=	20.000	Ohm	= 20 kOhm
204:	20	x	10.000	=	200.000	Ohm	= 200 kOhm
501:	50	x	10	=	500	Ohm	
502:	50	x	100	=	5.000	Ohm	= 5 kOhm
503:	50	x	1.000	=	50.000	Ohm	= 50 kOhm
504:	50	x	10.000	=	500.000	Ohm	= 500 kOhm

Leuchtdiode (LED)



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.

- 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 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.

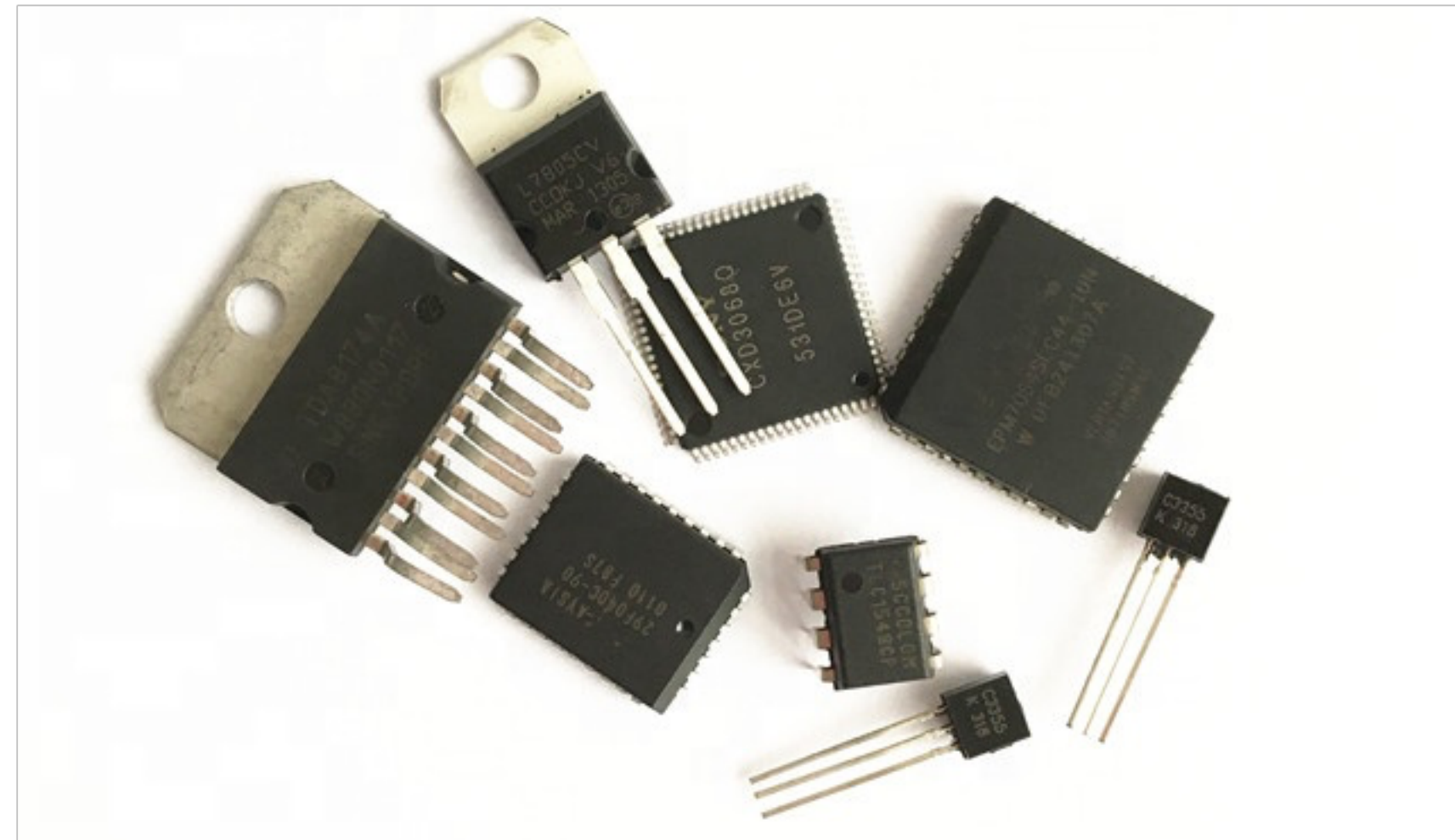
Bei einem deutlich kleineren Widerstand als 1 kOhm kann die LED kaputt gehen. Bei einem deutlich zu großen Widerstand als 20 kOhm leuchtet die LED zu schwach oder gar nicht.

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 (+).

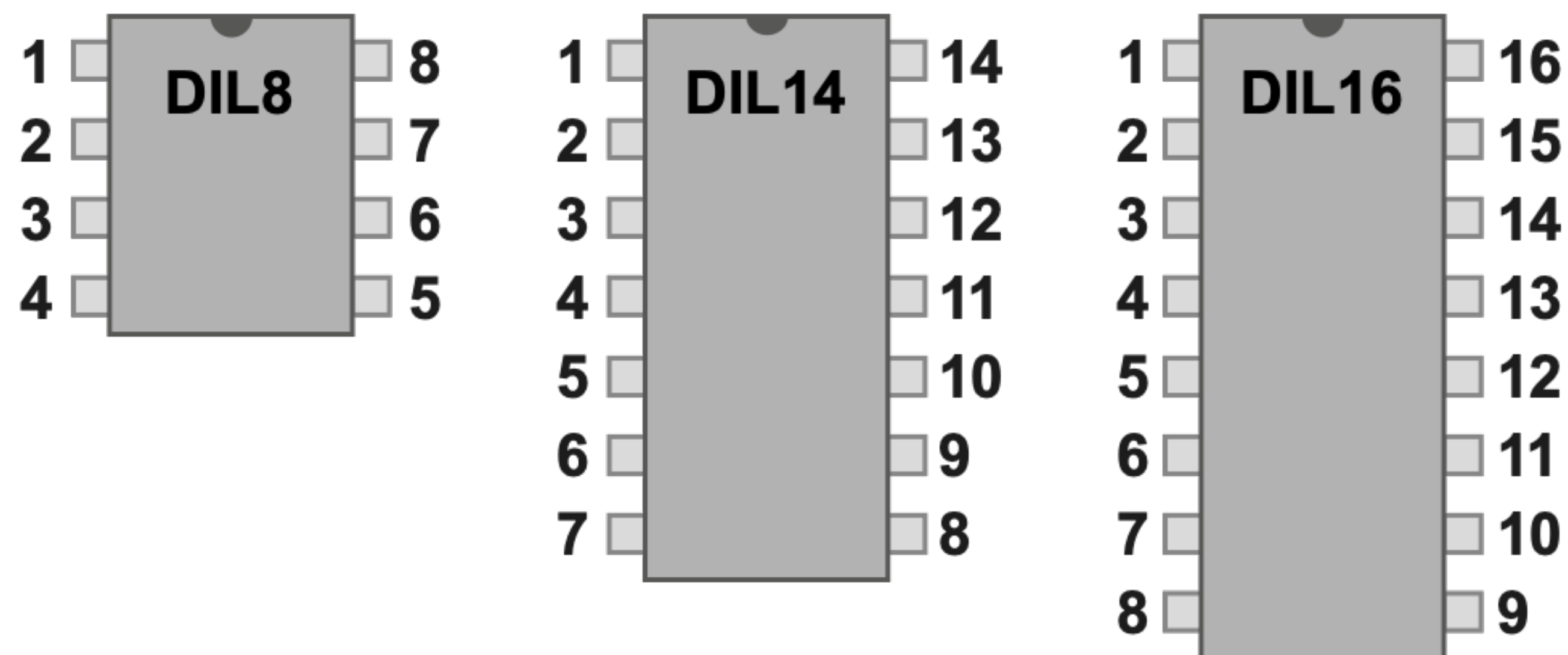
Einfach zu merken: Das Pluszeichen hat einen Strich mehr als das Minuszeichen und deshalb ist der Anschlussdraht der Anode etwas länger. Außerdem sind die meisten LEDs auf der Minusseite abgeflacht, wie eben ein Minus, oder das "K" der Kathode.

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)



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

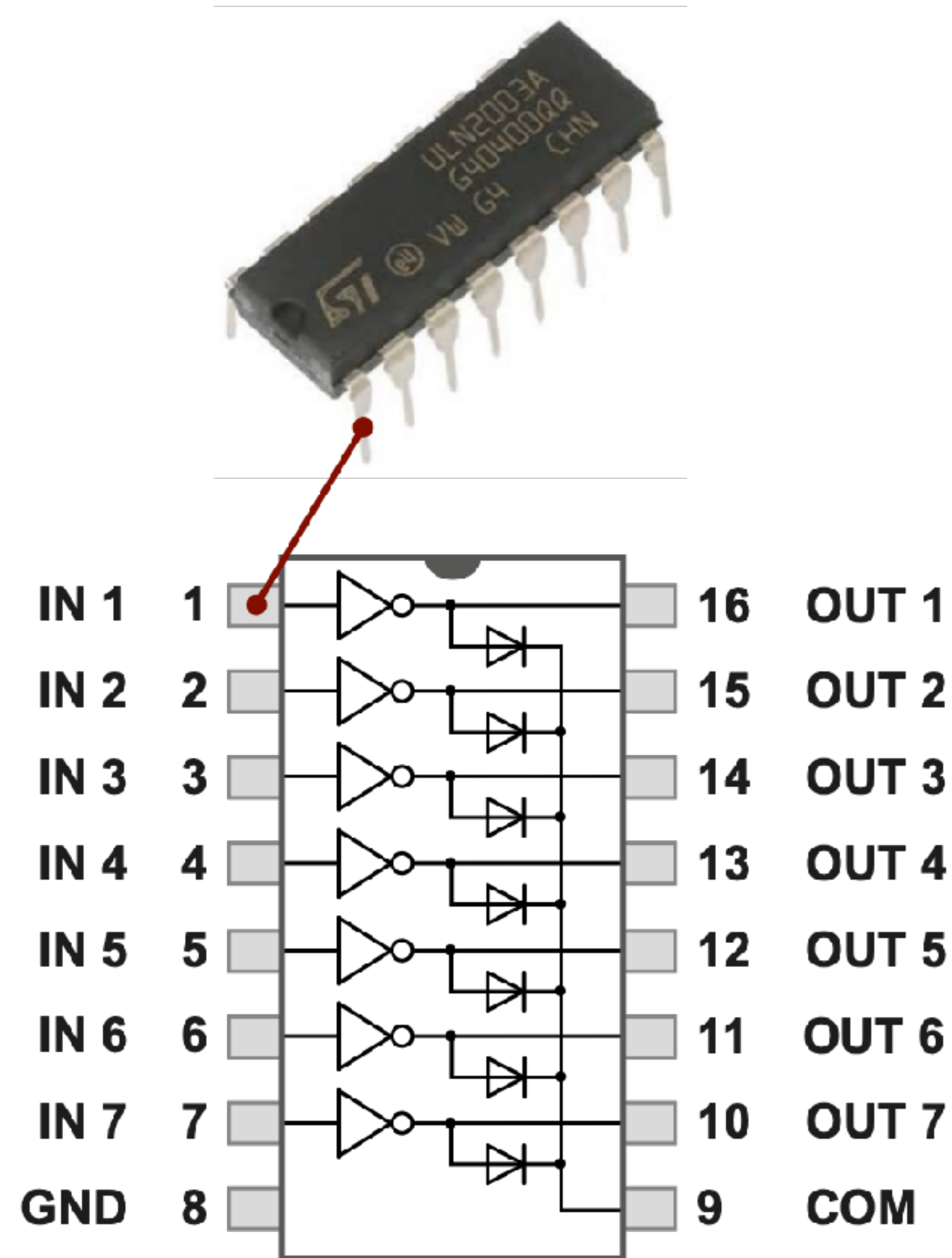


Wenn man auf die beschriftete Seite des ICs schaut, kann man an einer kurzen Seite eine Einkerbung entdecken. Diese Einkerbung kennzeichnet „oben“. Links von der Einkerbung beginnt die Nummerierung der Pins von 1 an nach unten gegen den Uhrzeigersinn zu zählen.

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 lötl- 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.

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.

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.

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 einem 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.

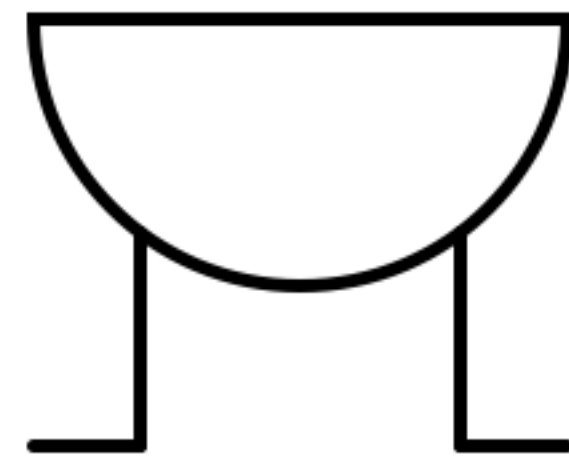
Masse bzw. Ground (GND)

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.

Summer (Buzzer)



Bei einem Summer muss man in der Regel auf die **Polarität** achten. Dabei ist der positive Pol mit einem Plus (+) oben auf dem Gehäuse gekennzeichnet. In der Regel ist das Anschlussbein beim Pluspol länger.

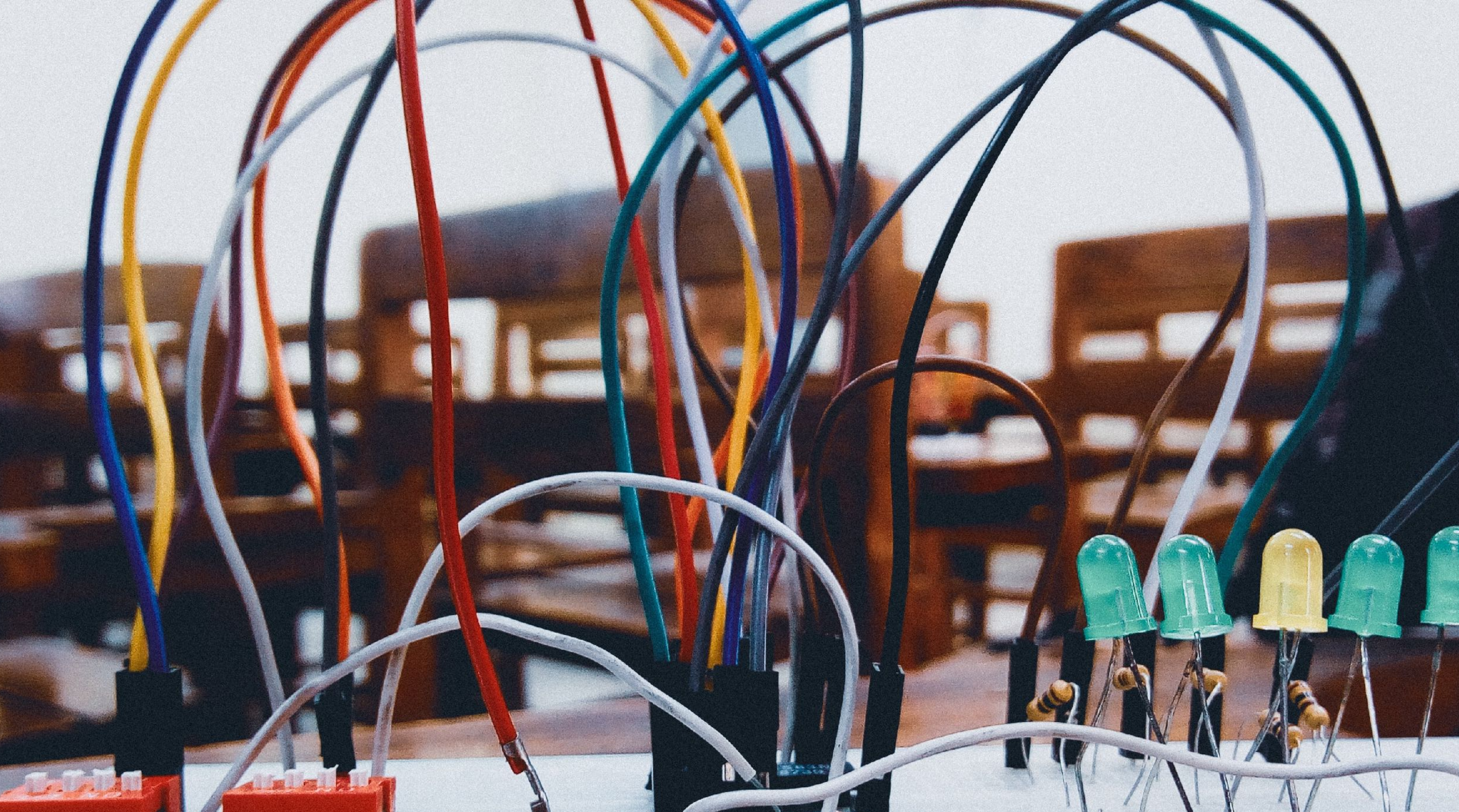
Die **Betriebsspannung** der marktüblichen Summer ist von der Farbe des bedruckten Etiketts abhängig. Die Farbe **Blau** deutet auf **5 Volt** hin. Die Betriebsspannung darf aber auch darüber oder darunter liegen.

- akustischer Signalgeber
- Tonausgabe mit festgelegter Frequenz (Tonhöhe)

Ein Summer, auch Buzzer genannt, ist ein akustischer Signalgeber. Er wandelt ein elektrisches Signal in ein akustisches Signal um. Das heißt, es erklingt ein Ton mit einer festgelegten Frequenz, die man nicht ändern kann. Über die Zeit gesehen, empfinden wir diesen Ton als nervig, weshalb er abschaltbar sein soll.

Ein Summer ist einfach zu benutzen. Er muss nur an eine Spannung angelegt werden und schon „summt“ er. Der Nachteil dabei, die Frequenz und damit die Tonhöhe können nicht beeinflusst werden. Die ist festgelegt. Desweiteren hat man nur wenig Einfluss auf die Lautstärke.

Das Etikett mit der Beschriftung „REMOVE SEAL AFTER WASHING“ dient zum Schutz des Summers. Nach der maschinellen Produktion werden Platinen „gewaschen“. Danach kann das Etikett entfernt werden.



Experimente und Anwendungen

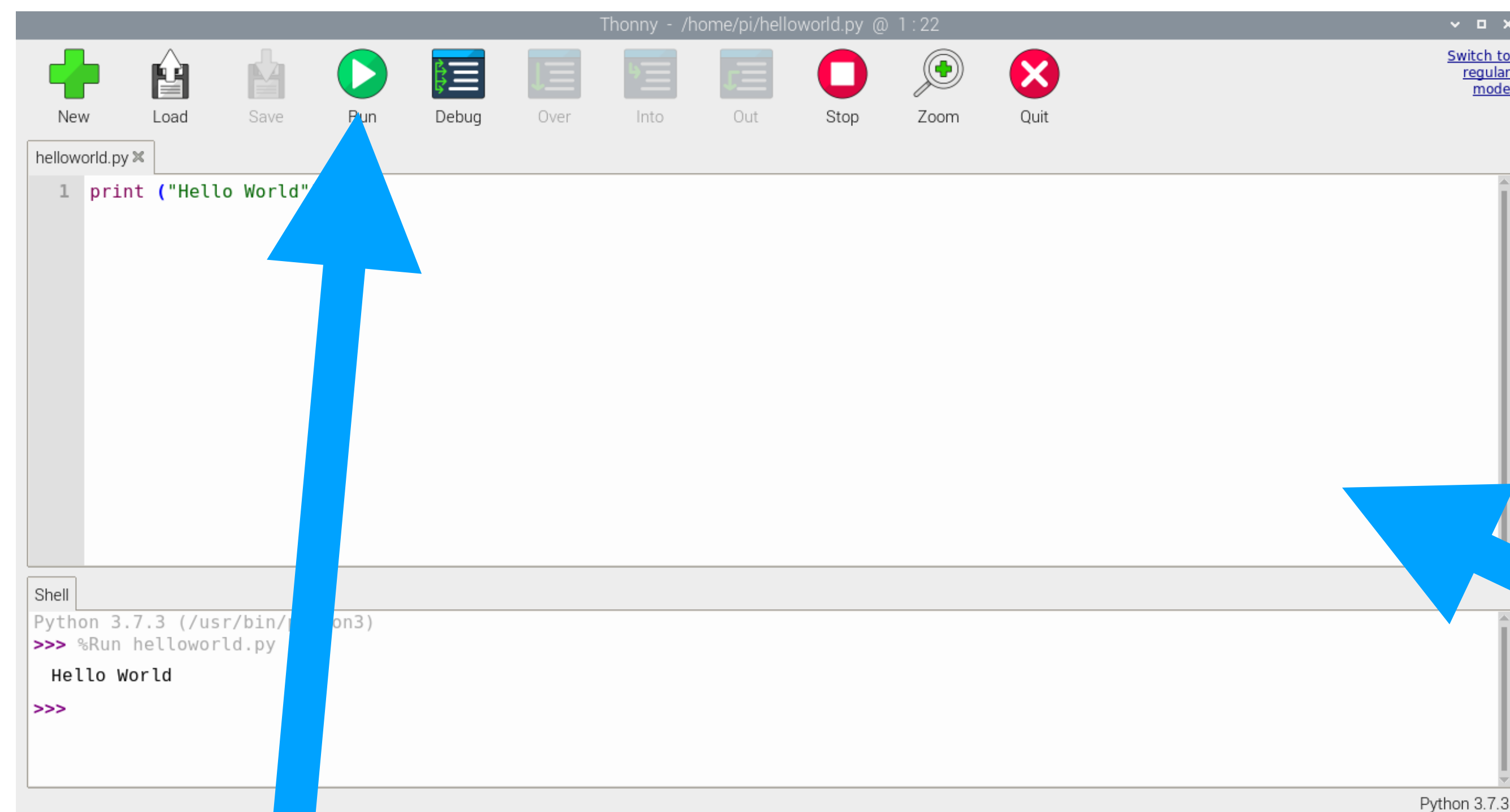
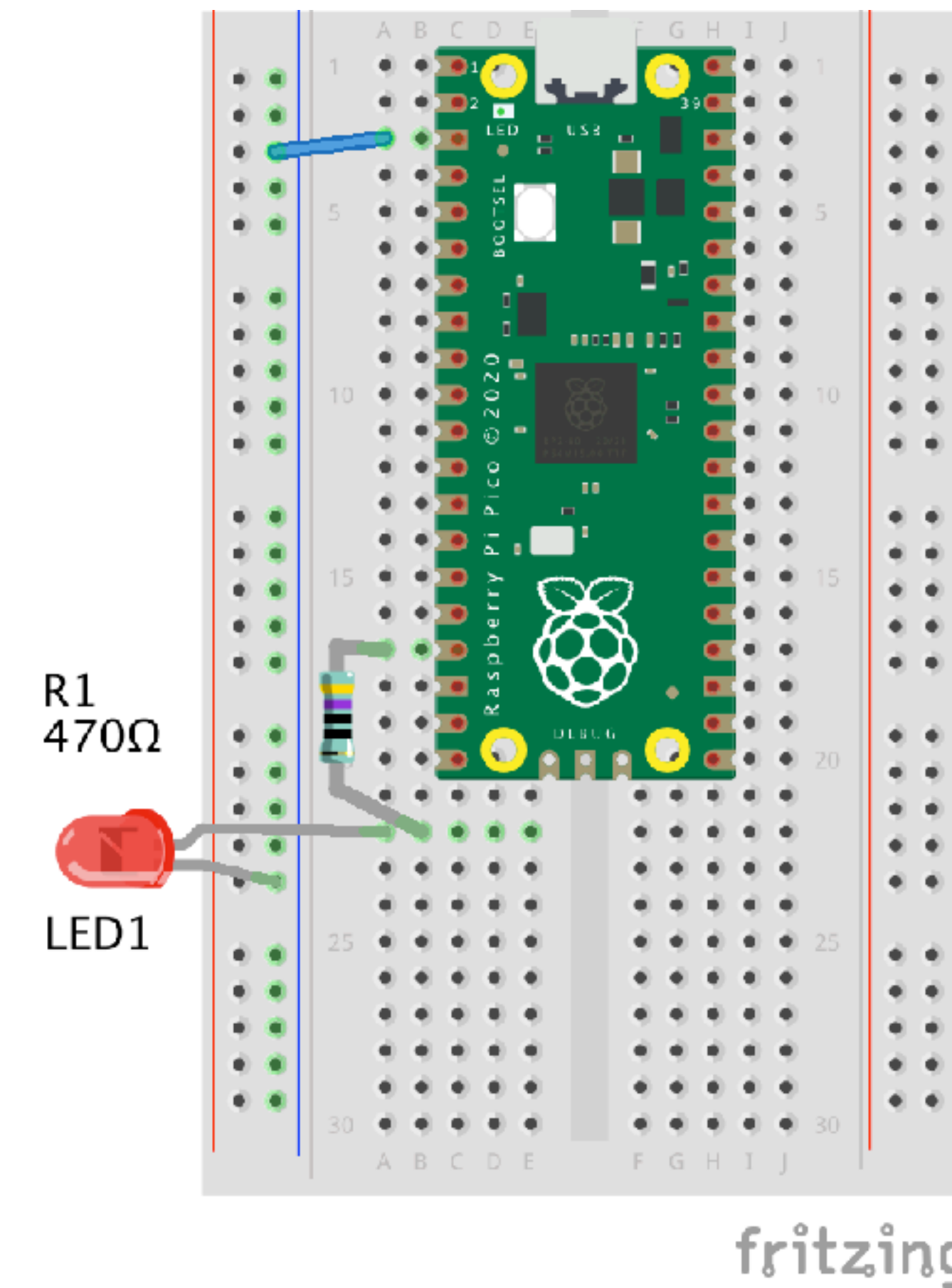
Experimente und Anwendungen mit MicroPython

- Vorgehensweise beim Programmieren (1)
- Vorgehensweise beim Programmieren (2)
- Online-Workshop: PicoBello
- Onboard-LED einschalten und ausschalten
- Onboard-LED blinken lassen
- Externe LED einschalten und ausschalten
- Externe LED blinken lassen
- LED-Helligkeit steuern
- LED mit Taster einschalten und ausschalten
- Taster-Zustand auswerten und mit einer LED anzeigen
- Temperatur mit dem integrierten Temperatursensor messen
- Raspberry Pi Pico W als WLAN-Client
- Statische IPv4-Konfiguration
- Internet-Verbindung prüfen
- Datum und Uhrzeit per NTP-Zeitserver einstellen
- Webserver-Betrieb
- E-Mail senden
- Online-Wetterstation mit Vorhersage
- MQTT-Publisher und MQTT-Subscriber

Hinweis: Weil die Beschreibungen und der Programmcode für die folgenden Anwendungen und Lösungen teilweise sehr umfangreich sein können, wird in manchen Beschreibungen auf eine Quelle auf unsere Webseite verlinkt. Das muss kein Nachteil sein. Dort findest Du auch Fehlerhinweise mit Lösungen, Tipps zur praktischen Nutzung und Ideen für Erweiterungen.

Vorgehensweise beim Programmieren (1)

1. Zuerst die Schaltung aufbauen



2. Dann den Programmcode ins Textfeld kopieren und speichern

```
from machine import Pin
from time import sleep

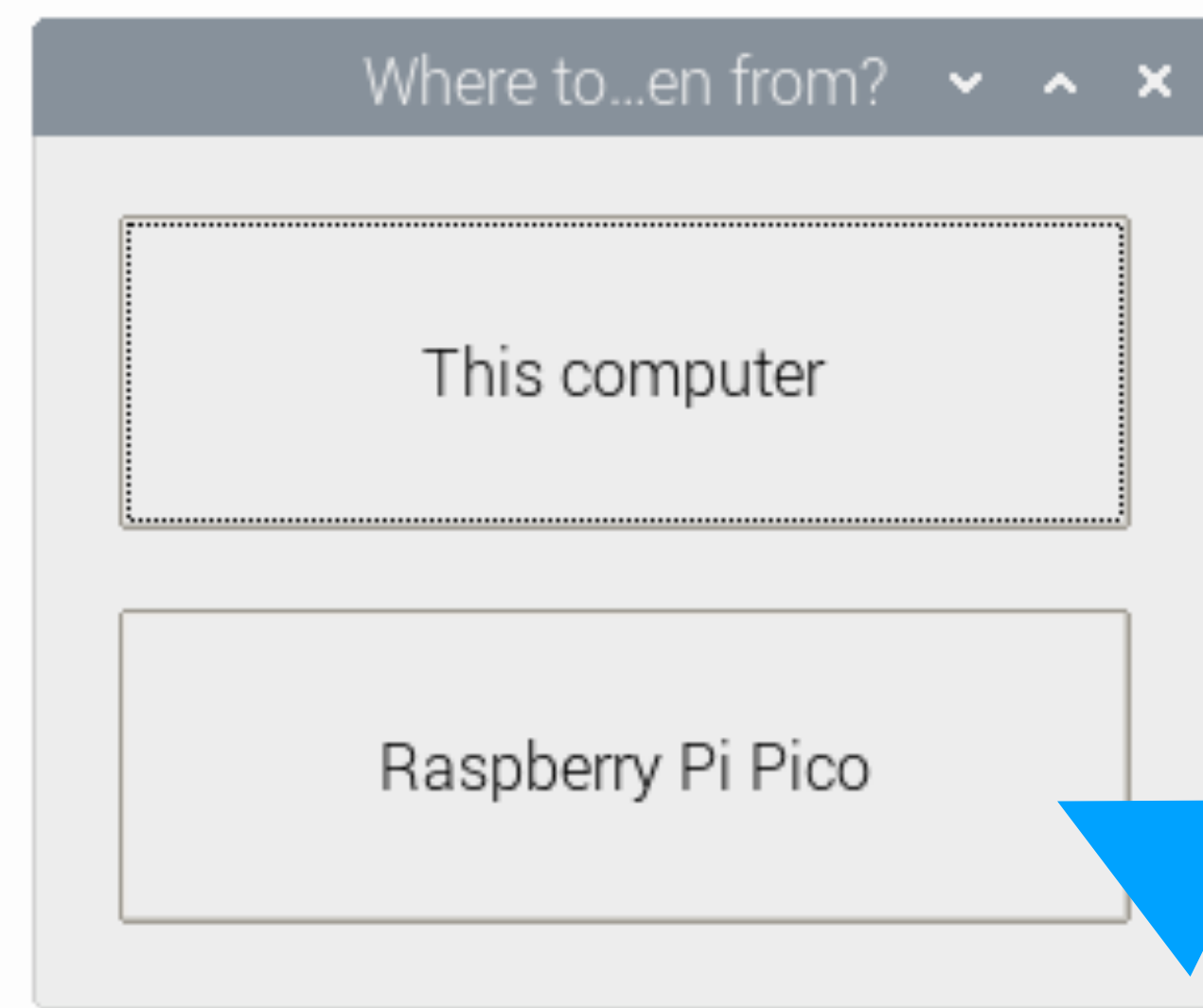
led = Pin('LED', Pin.OUT)

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

3. Danach das Programm ausführen

4. Funktion von Aufbau und Programm prüfen

Vorgehensweise beim Programmieren (2)



Programmcode mit der Dateiendung „.py“ auf dem Pico speichern.

1. Zuerst die Schaltung aufbauen. Zum Beispiel auf einem Steckbrett.
2. Im Thonny-Editor eine neue Datei öffnen, den Programmcode ins Textfeld kopieren und die Datei mit der Dateiendung „.py“ speichern und als Speicherort den Raspberry Pi Pico auswählen.
3. Das Programm ausführen bzw. starten.
4. Die Funktion des Programms prüfen.

Auch wenn ein Programm nicht mehr läuft, leuchten die Leuchtdioden immer noch, die beim letzten Programmschritt geleuchtet haben. Es bleiben also „Zustände“ im Speicher erhalten. Wenn man dann ein anderes Programm startet, dann leuchten manche LEDs einfach weiter.

Es empfiehlt sich, den Raspberry Pi Pico bei einem neuen Programm vorher einmal auszustecken und neu einzustecken. Dabei werden alle Rückstände des alten Programms aus dem Speicher gelöscht.

Hinweis: Nach dem erneuten Einstecken muss die Verbindung zum Raspberry Pi Pico durch Klicken auf „Stop“ zurückgesetzt werden.



```
Couldn't find the device automatically.  
Check the connection (making sure the device is not in bootloader mode) or choose  
"Configure interpreter" in the interpreter menu (bottom-right corner of the window)  
to select specific port or another interpreter.
```

Online-Workshop: Picobello

Du musst das
nicht alleine
machen!



Im Online-Workshop PicoBello ...

- ... Lernst Du Programmieren mit dem Raspberry Pi Pico.
- ... Wir arbeiten mit den Teilnehmern gemeinsam.
- ... Bekommst Du Unterstützung bei individuellen Problemen.

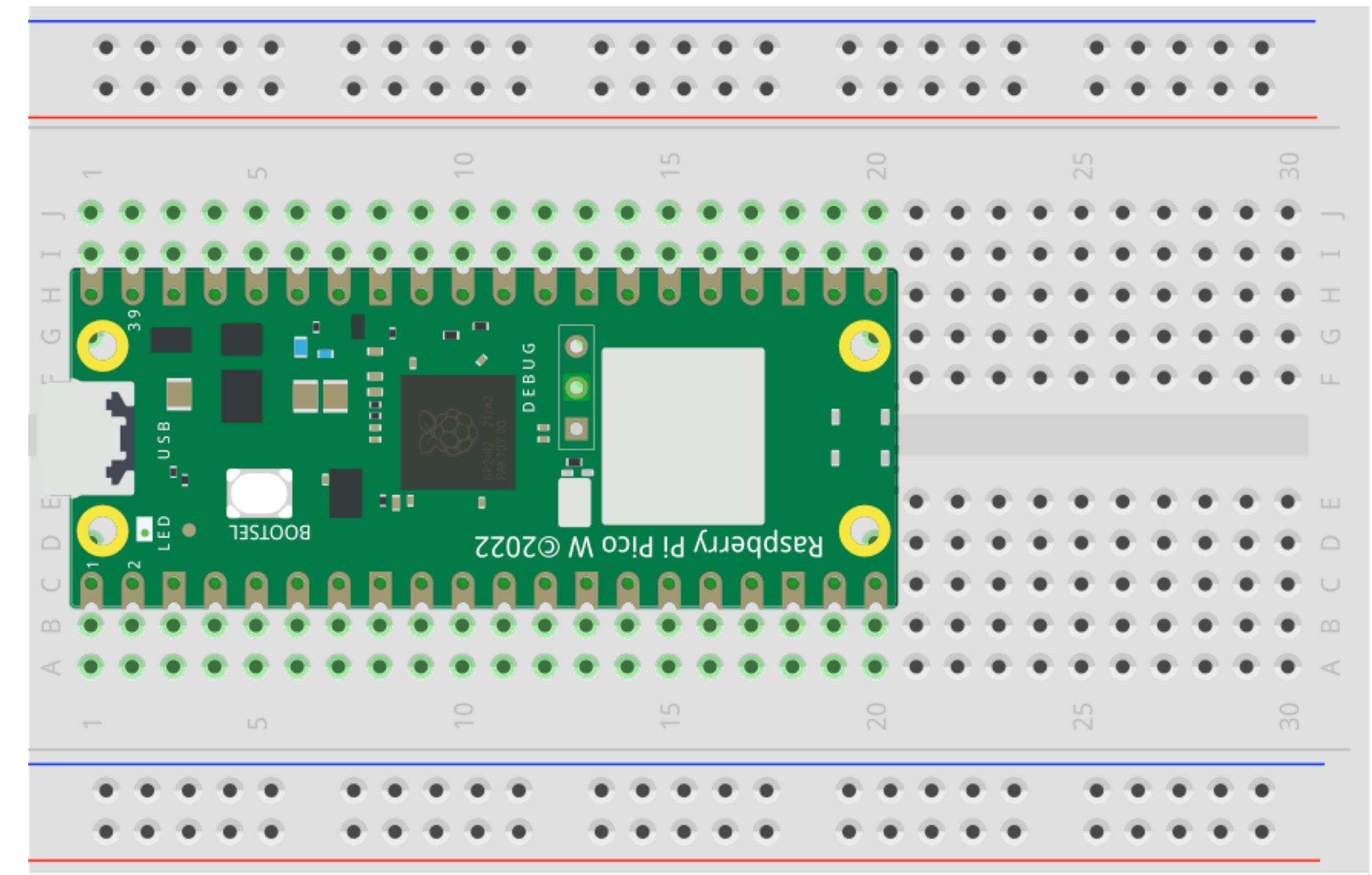
<https://www.elektronik-kompendium.de/service/events/>

Onboard-LED einschalten und ausschalten

Mit diesem Aufbau geht es erst einmal nur darum, den Raspberry Pi Pico W mit einem allerersten Programm auszuprobieren. Dafür benutzen wir die Onboard-LED des Picos. Damit kann man den Pico testen, prüfen und erste Experimente durchführen. Ohne den Aufwand einer externen Beschaltung.

Es geht nur darum, die Onboard-LED auf dem Raspberry Pi Pico W einzuschalten und wieder auszuschalten.

Der Vorteil dieses Aufbaus ist, dass außer dem Raspberry Pi Pico W nichts weiter erforderlich ist.



```
# Bibliotheken laden
from machine import Pin
from time import sleep

# Initialisierung der Onboard-LED
led_onboard = Pin('LED', Pin.OUT)

# LED einschalten
led_onboard.on()

# 5 Sekunden warten
sleep(5)

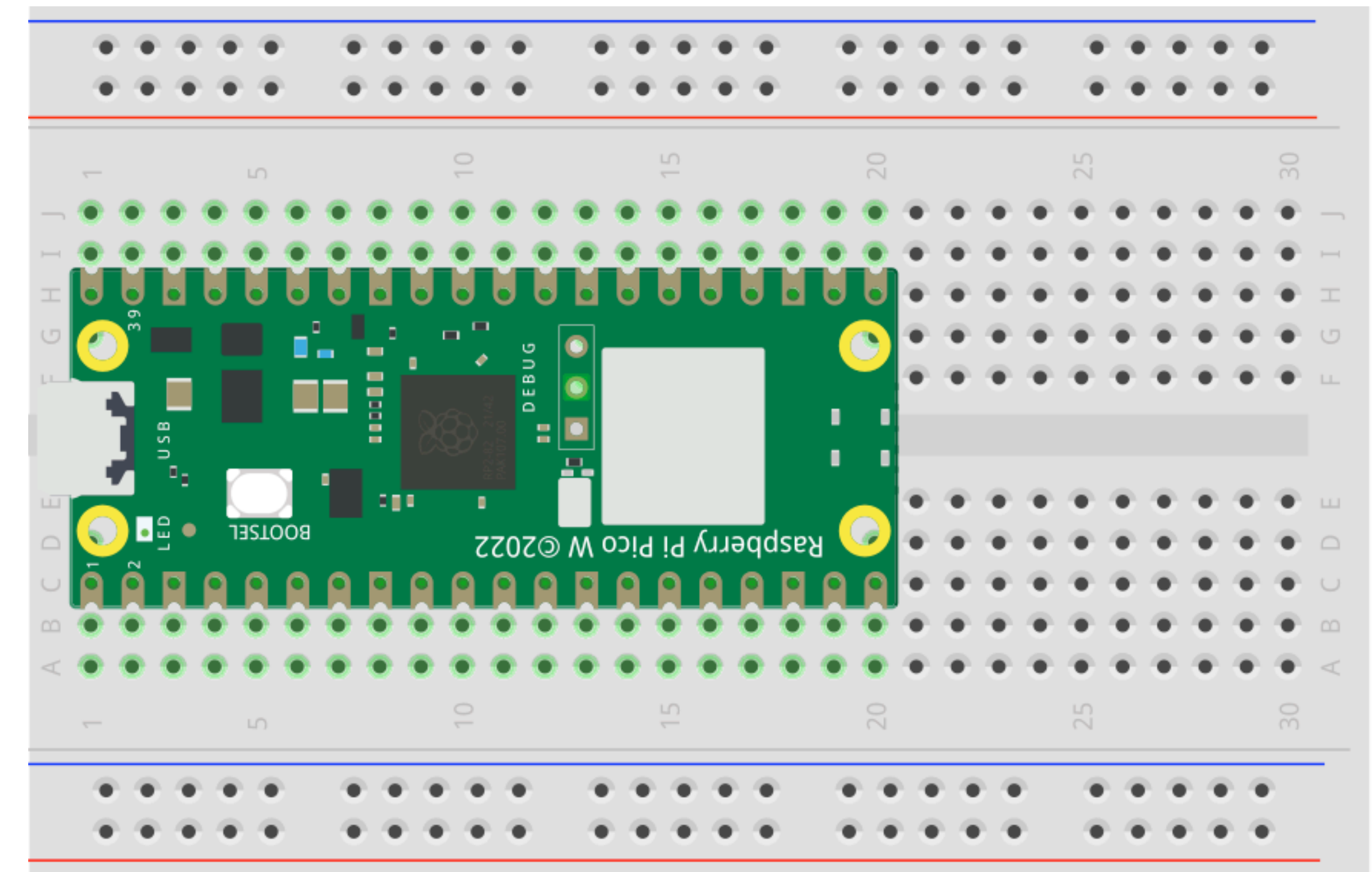
# LED ausschalten
led_onboard.off()
```

Onboard-LED blinken lassen (1)

Elektronik macht immer dann am meisten Spaß, wenn es blinkt und blitzt. Aber bitte nicht zu viel davon. Wenn es zu sehr blitzt, dann ist es meistens kaputt. Deshalb beschränken wir uns sicherheitshalber auf das Blinken einer Leuchtdiode.

Damit eine LED blinkt oder blitzt gibt es in der Elektronik ein paar Grundschaltungen. Die folgenden Programmcodes ersetzen die Funktion dieser elektronischen Schaltungen. Die Leuchtdiode wird per Software gesteuert, was uns viele Möglichkeiten der Steuerung eröffnet, während wir bei einer elektronischen Schaltung auf deren Funktion beschränkt bleiben. Mit MicroPython lässt sich das Blinken der Onboard-LED mit wenigen Zeilen erledigen.

Dabei gibt es nicht nur einen Weg, wie man die Onboard-LED zum Blinken bringen kann. Es gibt gleich mehrere davon. Und genau das sollst Du ausprobieren. Und es wäre denkbar, dass es noch weitere Lösungen gibt, die hier nicht berücksichtigt wurden.



Onboard-LED blinken lassen (2)

Nach dem Start der Programme beginnt die Onboard-LED zu blinken. Beim ersten Programm wird die LED nach einer bestimmten Wartezeit eingeschaltet und ausgeschaltet.

Wichtig: Beim Kopieren oder Abtippen auf die Einrückungen (4 Leerzeichen) achten.



Das zweite Programm nutzt die Toggle-Funktion, die zwischen zwei Zuständen hin- und herwechselt, unabhängig davon, welcher Zustand gerade herrscht.

```
from machine import Pin
from time import sleep

# Initialisierung der Onboard-LED
led_onboard = Pin('LED', Pin.OUT)

# Wiederholung (Endlos-Schleife)
while True:
    # LED einschalten
    led_onboard.on()
    # halbe Sekunde warten
    sleep(0.5)
    # LED ausschalten
    led_onboard.off()
    # 1 Sekunde warten
    sleep(1)
```

```
from machine import Pin
from time import sleep

# Initialisierung der Onboard-LED
led_onboard = Pin('LED', Pin.OUT)

# Wiederholung (Endlos-Schleife)
while True:
    # LED-Zustand wechseln (EIN/AUS)
    led_onboard.toggle()
    # 1 Sekunde warten
    sleep(1)
```

Onboard-LED blinken lassen (3)

Die beiden vorherigen Programme sind sicherlich zweckmäßig. Allerdings entspricht die Lösung mit einer Schleife eher dem Stil eines Anfängers. Wir wollen aber etwas lernen und auch unsere Programmierfähigkeiten verbessern. Deshalb nutzen wir hier die Lösung mit einem Timer. Das ist eine interne Funktion des Mikrocontrollers.

Zu beachten ist hier, dass der Wechsel von ein nach aus nicht über die Zeit, sondern die Frequenz in Hertz (Hz) eingestellt wird. Eine Frequenz gibt die Zahl von etwas an, das in einer Sekunde durchlaufen wird. Soll eine Leuchtdiode 2 mal in einer Sekunde blinken, dann gibt man 2 Hertz an.

Die Timer-Lösung hat den Vorteil, dass sie unabhängig von anderen Programmabläufen funktioniert. Bei einer Schleifen-Lösung blinkt die LED nur dann, wenn die Schleife nicht unterbrochen oder gestoppt wird.

```
# Bibliotheken laden
from machine import Pin, Timer

# Initialisierung der Onboard-LED
led_onboard = Pin('LED', Pin.OUT)

# Definition einer Funktion
def blink(timer):
    led_onboard.toggle()

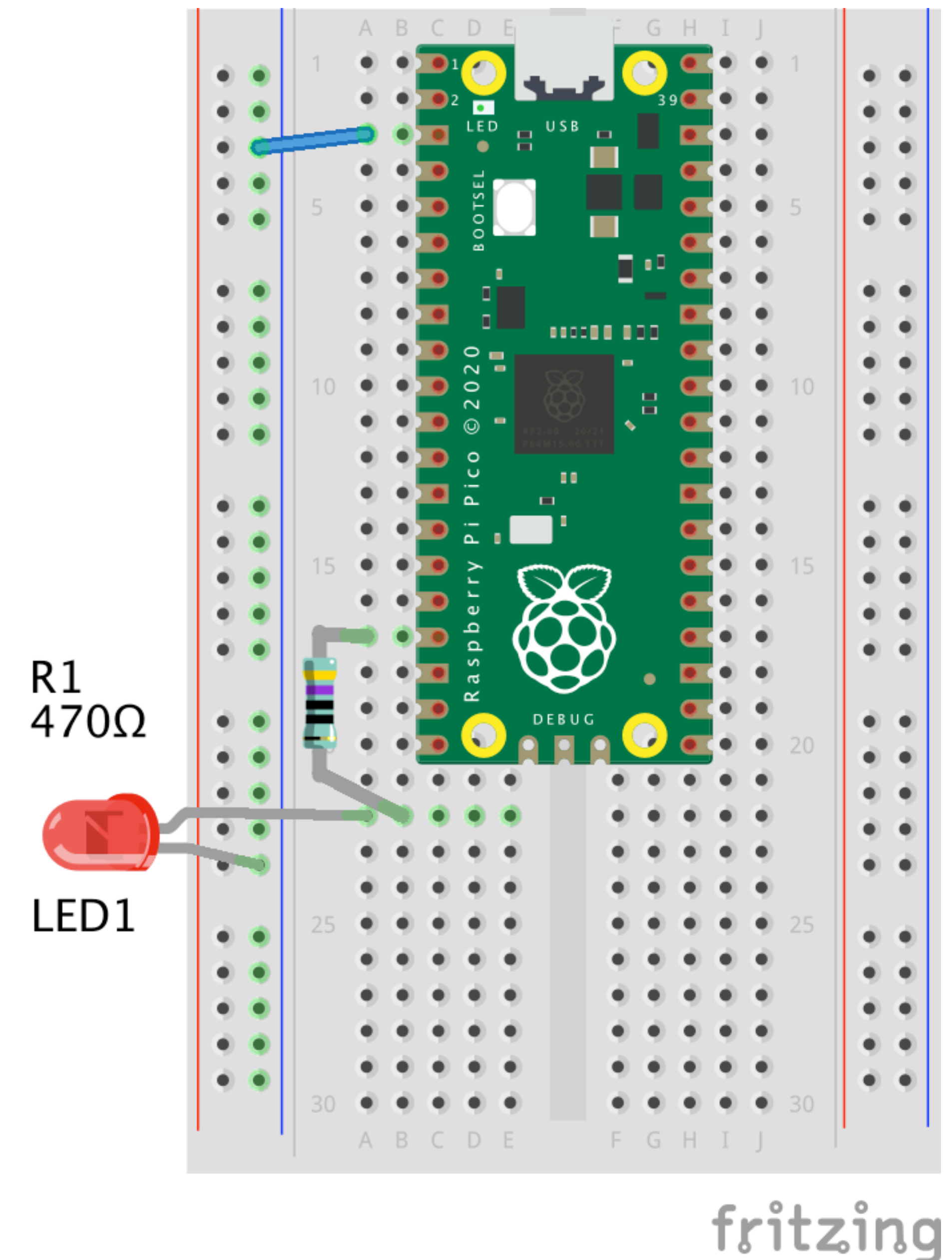
# Timer initialisieren
Timer().init(freq=2.5, callback=blink)
```


Externe LED einschalten und ausschalten (1)

In diesem Aufbau geht es darum, an einem Raspberry Pi Pico eine LED richtig anzuschließen und mit einem Programm die LED einzuschalten und wieder auszuschalten. Das ist nicht weiter schwer. Wir brauchen dazu nur eine LED (egal welche Farbe) und einen Widerstand. Auf einem Steckbrett werden die Bauteile zusammen mit den GPIO-Pins verbunden. Es bietet sich an, auch mal andersfarbige Leuchtdioden auszuprobieren.

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 in MicroPython für eine externe Beschaltung zu schreiben.

Wenn Du das alles geschafft hast, dann können Dich die nächsten Experimente nicht mehr schrecken.



LED1: Leuchtdiode, rot, gelb oder grün

R1: Widerstand, 470 Ohm (Gelb-Violett-Schwarz-Schwarz)

Externe LED einschalten und ausschalten (2)

Nach dem Start des Programms leuchtet die Leuchtdiode für 5 Sekunden. Dann geht sie wieder aus.

```
# Bibliotheken laden
from machine import Pin
from time import sleep

# Initialisierung von GPIO13 als Ausgang
led = Pin(13, Pin.OUT)

# LED einschalten
led.on()

# 5 Sekunden warten
sleep(5)

# LED ausschalten
led.off()
```

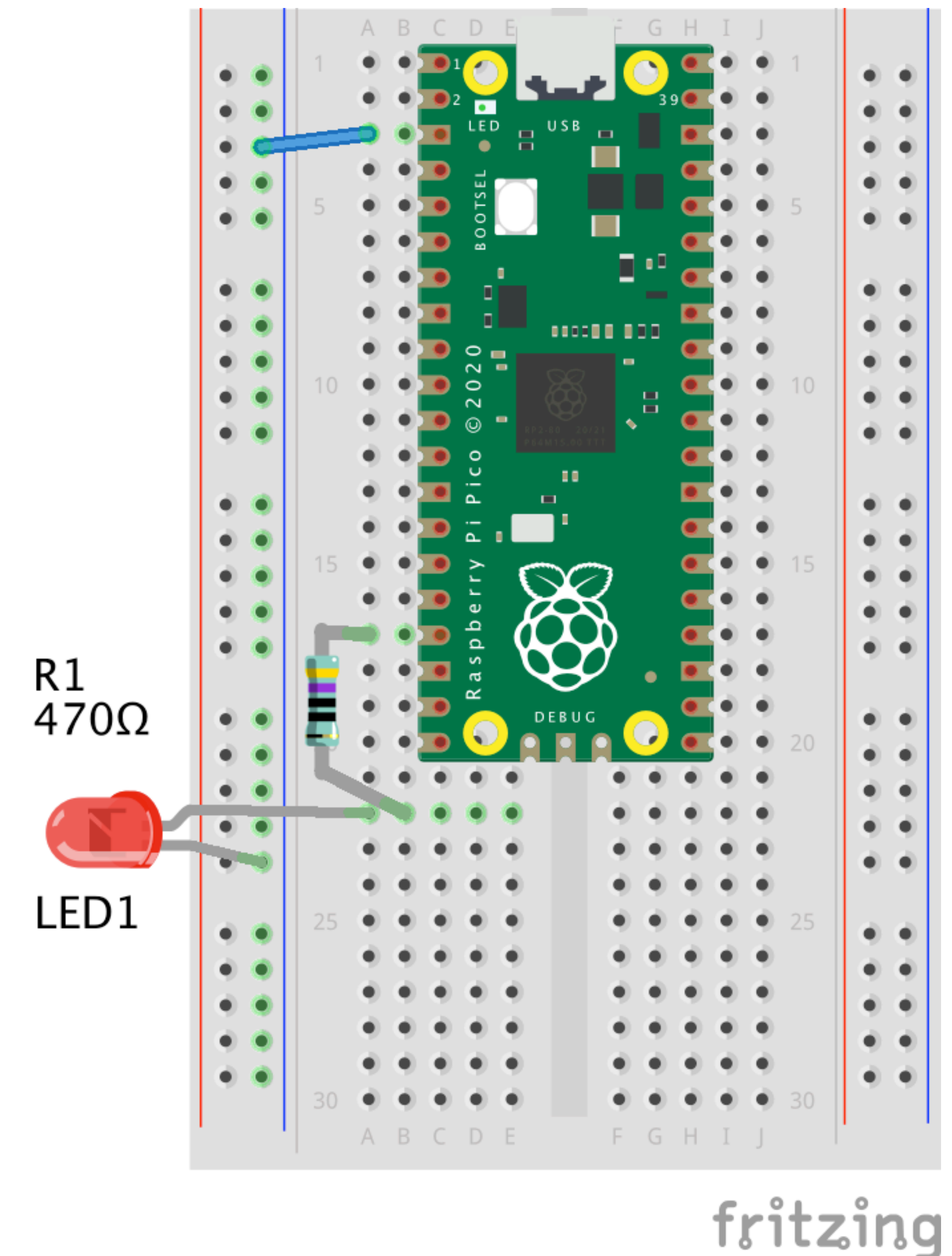
Externe LED blinken lassen (1)

Am Besten ist Elektronik immer dann, wenn es blinkt und blitzt. Aber wenn es zu sehr blitzt, ist danach meistens etwas kaputt. Deshalb beschränken wir uns sicherheitshalber auf das Blinken einer Leuchtdiode. Für den Raspberry Pi Pico lässt sich in MicroPython das Blinken einer Leuchtdiode in wenigen Zeilen schreiben.

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 die Funktion dieser Grundschaltungen. Das einzige, was an Hardware bleibt, ist die Leuchtdiode mit dem Vorwiderstand. Und natürlich der Raspberry Pi Pico.

Die Leuchtdiode wird per Software gesteuert.

Es gibt nicht nur einen Weg, wie man eine Leuchtdiode zum Blinken bringen kann. Es gibt gleich mehrere Möglichkeiten. Und genau die solltest Du alle ausprobieren. Und es wäre denkbar, dass es noch weitere Lösungen gibt, die hier nicht berücksichtigt wurden.



LED1: Leuchtdiode, rot, gelb oder grün

R1: Widerstand, 470 Ohm (Gelb-Violett-Schwarz-Schwarz)

Externe LED blinken lassen (2)

Nach dem Start der Programme beginnt die LED zu blinken. Beim ersten Programm wird die LED nach einer bestimmten Wartezeit eingeschaltet und ausgeschaltet.

Das zweite Programm nutzt die Toggle-Funktion, die zwischen zwei Zuständen hin- und herwechselt, unabhängig davon, welcher Zustand gerade herrscht.

```
from machine import Pin
from time import sleep

# Initialisierung von GPIO13 als Ausgang
led = Pin(13, Pin.OUT)

# Wiederholung (Endlos-Schleife)
while True:
    # LED einschalten
    led.on()
    # halbe Sekunde warten
    sleep(0.5)
    # LED ausschalten
    led.off()
    # 1 Sekunde warten
    sleep(1)
```

```
from machine import Pin
from time import sleep

# Initialisierung von GPIO13 als Ausgang
led = Pin(13, Pin.OUT)

# Wiederholung (Endlos-Schleife)
while True:
    # LED-Zustand wechseln (EIN/AUS)
    led.toggle()
    # 1 Sekunde warten
    sleep(1)
```

Externe LED blinken lassen (3)

Die beiden vorherigen Programme sind sicherlich zweckmäßig. Allerdings entspricht die Lösung mit einer Schleife eher dem Stil eines Anfängers. Wir wollen aber hier etwas lernen und auch unsere Programmierfähigkeiten verbessern. Deshalb nutzen wir hier die Lösung mit einem Timer. Das ist eine interne Funktion des Mikrocontrollers.

Zu beachten ist hier, dass der Wechsel von ein nach aus nicht über die Zeit, sondern die Frequenz in Hertz (Hz) eingestellt wird. Eine Frequenz gibt die Zahl von etwas an, das in einer Sekunde durchlaufen wird. Soll eine Leuchtdiode 2 mal in einer Sekunde blinken, dann gibt man 2 Hertz an.

Die Timer-Lösung hat den Vorteil, dass sie unabhängig von anderen Programmabläufen funktioniert. Bei einer Schleifen-Lösung blinkt die LED nur dann, wenn die Schleife nicht unterbrochen oder gestoppt wird.

```
# Bibliotheken laden
from machine import Pin, Timer

# Initialisierung von GPIO13 als Ausgang
led = Pin(13, Pin.OUT)

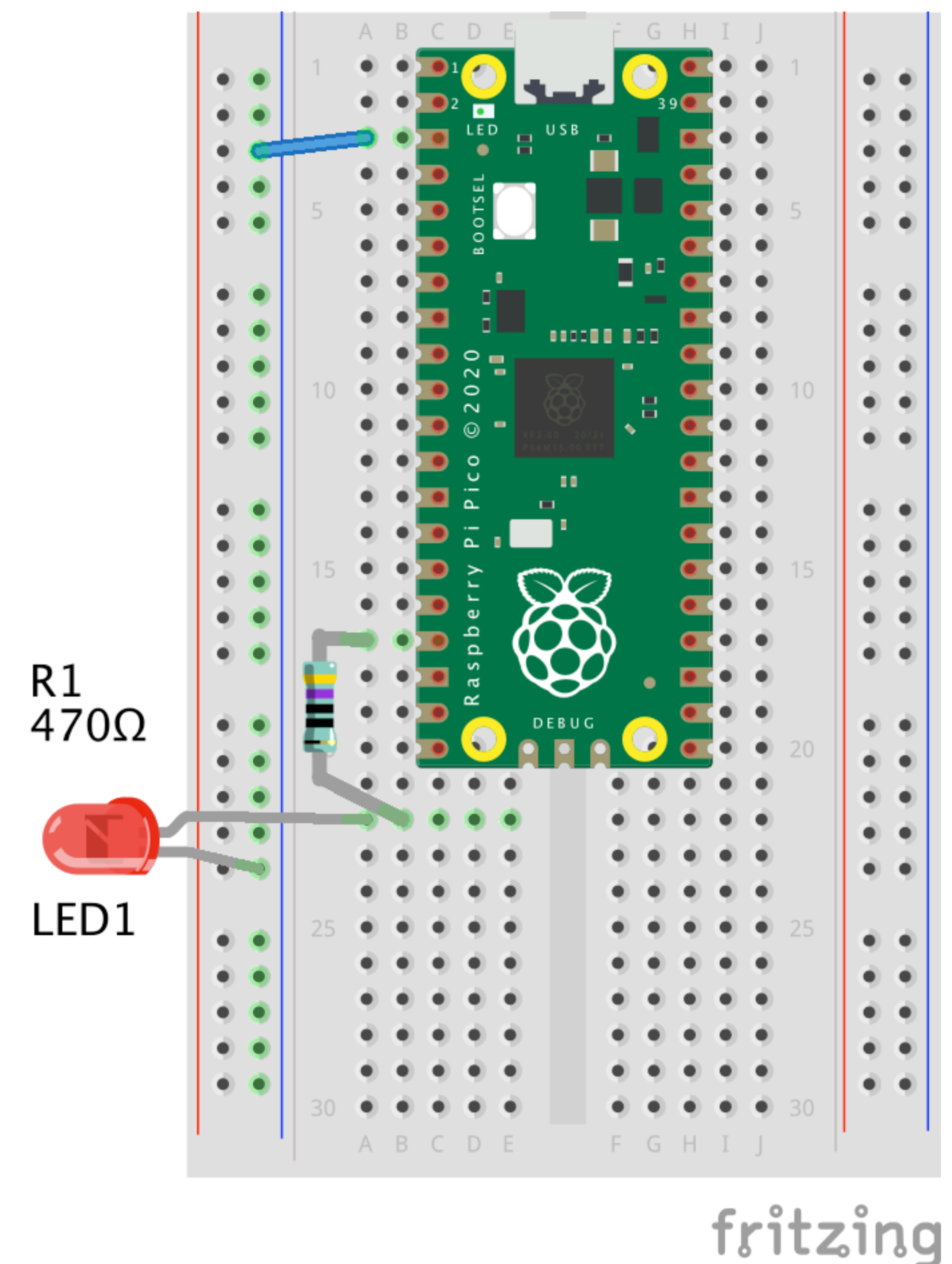
# Definition einer Funktion
def blink(timer):
    led.toggle()

# Timer initialisieren
Timer().init(freq=2.5, callback=blink)
```

LED-Helligkeit steuern (1)

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 ausgegeben 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.



LED1: Leuchtdiode, rot, gelb oder grün
R1: Widerstand, 470 Ohm (Gelb-Violett-Schwarz-Schwarz)

LED-Helligkeit steuern (2)

Im Programmcode wird der GPIO für die Leuchtdiode als PWM-Ausgang initialisiert und eine Frequenz von 1.000 Hertz (Hz) eingestellt. Anschließend wird innerhalb der Schleife das Duty-Cycle, das Verhältnis zwischen Impuls und Pause des Signals erhöht. Die Leuchtdiode leuchtet also immer heller, geht aus und wird wieder heller.

Verschiedene Parameter im Programm sorgen dafür, dass die LED pulsiert. Der Programmcode lädt zum Experimentieren ein.

- Die Zeile „sleep(0.1)“ gibt an wie lange in Sekunden zwischen den Veränderungsschritten gewartet werden soll. Vergrößere und verkleinere den Wert von „0.1“. Zum Beispiel auf „0.5“ oder „0.05“.
- Softwareseitig kann das PWM-Signal einen Wert zwischen 0 und 65.535 annehmen. Bei 0 ist die LED aus. Bei 65.535 leuchtet die LED am hellsten. Allerdings leuchtet die LED schon wesentlich vor 65.535 sehr hell. Verkleinere den Wert von „65535“ so weit, wie es wirklich nötig ist.
- Von 0 (LED aus) nach 65.535 (LED leuchtet voll) wird in 3000er Schritten die Helligkeit gesteigert. Vergrößere den Wert von „3000“ und verkleinere ihn, um den Effekt auf die Helligkeitsänderung herauszufinden.

```
# Bibliotheken laden
from machine import Pin, PWM
from time import sleep

# Initialisierung von GPIO13 als PWM-Ausgang
led = PWM(Pin(13))

# PWM-Einstellung: Frequenz in Hertz (Hz)
led.freq(1000)

i = 0

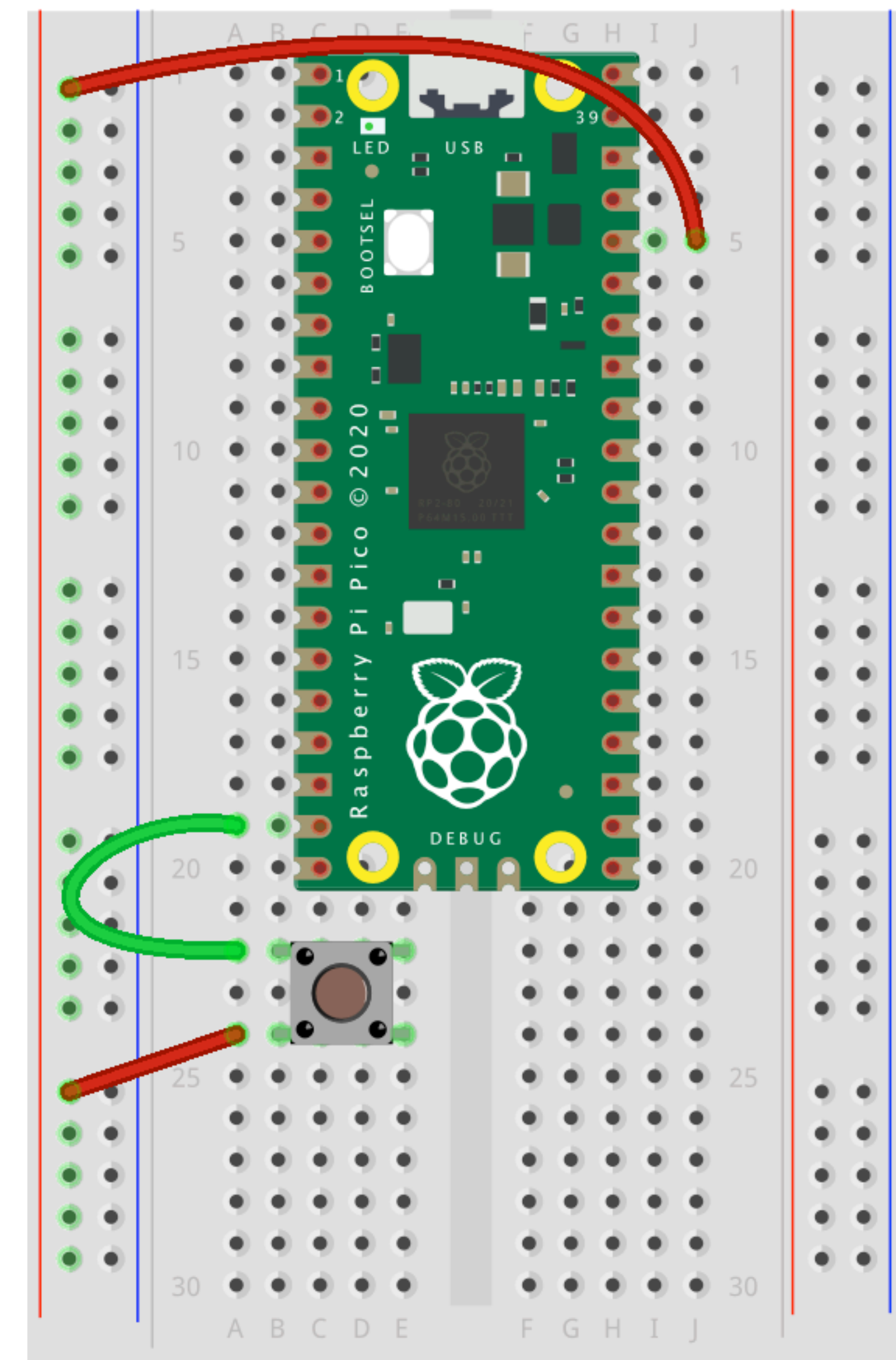
# Wiederholung (Endlos-Schleife)
while True:
    led.duty_u16(i)
    sleep(0.1)
    i = i + 3000
    if i > 65535:
        i = 0
```

Taster-Zustand auswerten und mit einer LED anzeigen (1)

Ein Taster kann zwei Zustände haben: „gedrückt“ und „nicht gedrückt“. Entsprechend für „Ein“ und „Aus“. Das entspricht der binären Logik und ist eigentlich ganz einfach. Doch zu allem Überfluss gibt es gleich 4 Möglichkeiten, wie man einen Taster mit dem Raspberry Pi Pico verbindet und wie der GPIO-Eingang initialisiert werden muss.

Zum Anzeigen des Taster-Zustands verwenden wir die Onboard-LED des Picos.

Das folgende Programmbeispiele berücksichtigt völlig wertfrei nur eine Variante. Denkbar wäre es, dass in der Praxis eine andere Variante besser wäre.



fritzing

Der Taster wird auf dem Steckbrett auf der einen Seite mit +3,3V (VCC) und auf der anderen Seite mit dem GPIO verbunden.

Taster-Zustand auswerten und mit einer LED anzeigen (2)

Die Funktion des Programms ist denkbar einfach. Ein GPIO wird als Ausgang definiert. Ein GPIO wird als Eingang definiert, an dem der Taster angeschlossen ist. Wird der Taster gedrückt, dann leuchtet die Leuchtdiode auf dem Pico. Wird der Taster losgelassen, dann wird sie wieder ausgeschaltet.

```
# Bibliotheken laden
from machine import Pin

# Initialisierung der Onboard-LED
led_onboard = Pin('LED', Pin.OUT)

# Initialisierung von GPIO14 als Eingang
btn = Pin(14, Pin.IN, Pin.PULL_DOWN)

# Funktion zur Taster-Auswertung
while True:
    if btn.value() == 1:
        led_onboard.on()
    else:
        led_onboard.off()
```

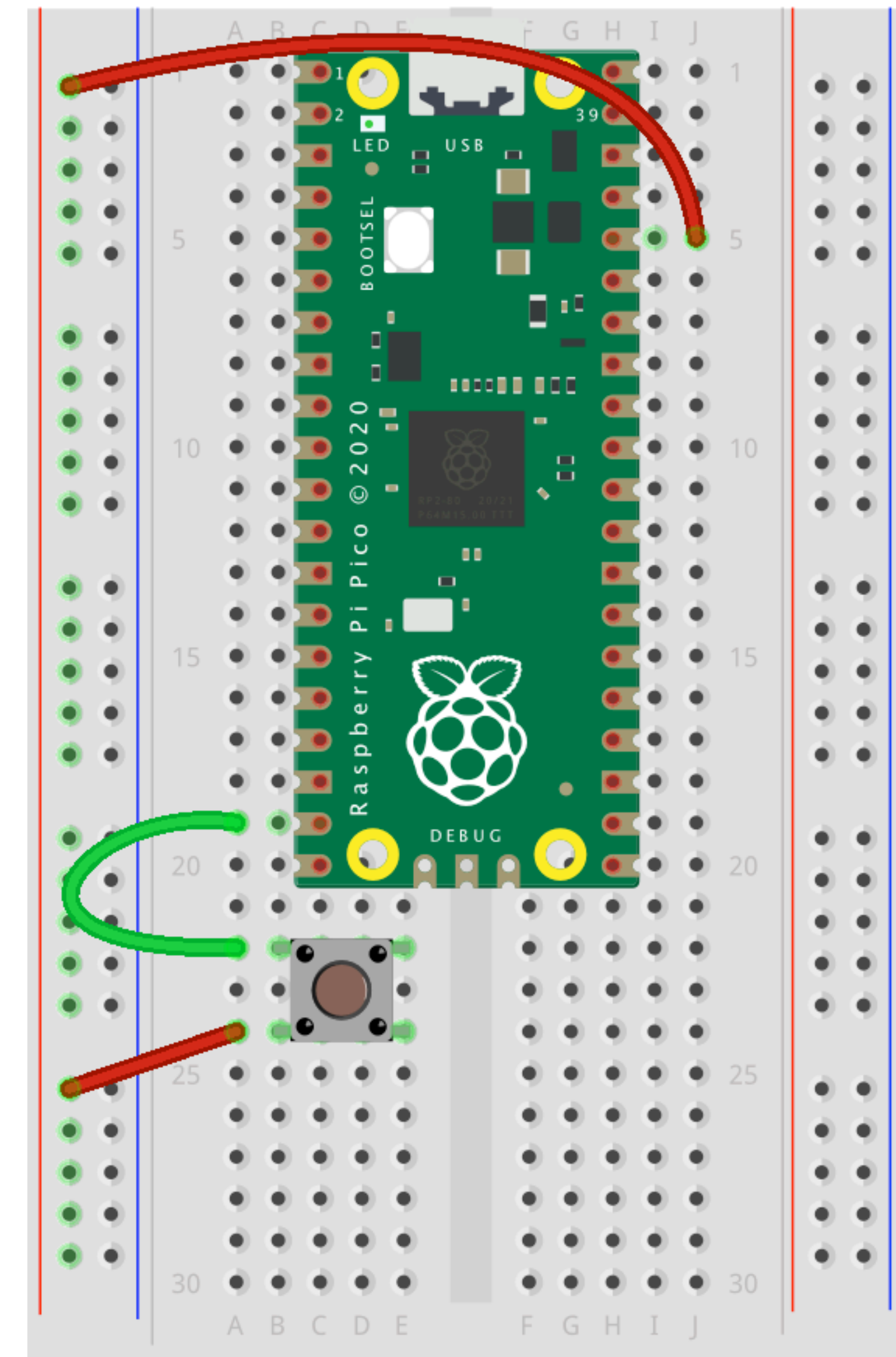
LED mit Taster einschalten und ausschalten (1)

Damit eine Leuchtdiode (LED) beim Drücken eines Tasters an- 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. Wenn die Funktion fest verdrahtet oder gelötet ist, dann ist eine Änderung nicht so einfach möglich. Soll der Taster eine andere Funktion haben, dann muss man die Hardware ändern.

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.

Das folgende Programm hat eine Haltefunktion. Das heißt, drücken wir den Taster, dann wird die LED eingeschaltet. Drücken wir ein weiteres Mal wird die LED ausgeschaltet. Man spricht hier auch von einer manuellen Toggle-Funktion.



fritzing

Der Taster wird auf dem Steckbrett auf der einen Seite mit +3,3V (VCC) und auf der anderen Seite mit dem GPIO verbunden.

LED mit Taster einschalten und ausschalten (2)

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

Das besondere an diesem Programm ist, dass der Taster-Druck einen Interrupt auslöst und dadurch eine bestimmte Funktion im Programm ausführt.

Na, funktioniert es? Naja, irgendwo schon. Aber irgendwie auch nicht immer. Manchmal geht die LED an und gleich wieder aus. Die Funktion ist also nicht so ganz zuverlässig.

Doch woran liegt das? Das Problem sind die Taster. Die neigen zum Prellen. Gemeint ist, dass ein Taster bei Betätigung nicht zwangsläufig einen dauerhaften Kontakt herstellt, sondern durch den mechanischen Aufschlag der Kontakte zurückfedern kann. Das heißt, das Drücken eines Tasters kann zu einer mehrmaligen Kontakt- und damit Funktionsauslösung führen. Das sieht dann so aus, dass die LED angeht, aber auch sofort wieder ausgeht. Der eine Druck auf den Taster hat den internen Kontakt zwei oder mehrmals betätigt und den Interrupt zwei oder mehr ausgelöst.

```
# Bibliotheken laden
from machine import Pin

# Initialisierung der Onboard-LED
led_onboard = Pin('LED', Pin.OUT, value=0)

# Initialisierung von GPIO14 als Eingang
btn = Pin(14, Pin.IN, Pin.PULL_DOWN)

# Taster-Funktion
def button(pin):
    led_onboard.toggle()

# Taster-Auslösung
btn.irq(trigger=Pin.IRQ_RISING, handler=button)
```

LED mit Taster einschalten und ausschalten (3)

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

Das besondere an diesem Programm ist, dass der Taster-Druck einen Interrupt auslöst und dadurch eine bestimmte Funktion im Programm ausführt. Diese Funktion setzt erst einen Timer mit einer Verzögerung von 200 ms und führt dann die Funktion einmalig aus, die die LED ansteuert.

Diese Konstruktion im Programmcode führt dazu, dass wenn der Taster innerhalb der 200 ms prellt, also zwischen High und Low hin- und herschwingt, die Funktion nicht erneut ausgeführt wird.

Hinweis: Die hier dargestellt Lösung ist vergleichsweise einfach und nicht in jedem Fall die beste Lösung.

```
# Bibliotheken laden
from machine import Pin, Timer

# Initialisierung der Onboard-LED
led_onboard = Pin('LED', Pin.OUT, value=0)

# Initialisierung von GPIO14 als Eingang
btn = Pin(14, Pin.IN, Pin.PULL_DOWN)

# Timer erstellen
timer = Timer()

# Taster-Funktion
def on_pressed(timer):
    led_onboard.toggle()
    print('pressed')

# Entprell-Funktion
def debounce(pin):
    # Timer setzen (200 ms)
    timer.init(mode=Timer.ONE_SHOT, period=200, callback=on_pressed)

# Taster-Auslösung
btn.irq(handler=debounce, trigger=Pin.IRQ_RISING)
```

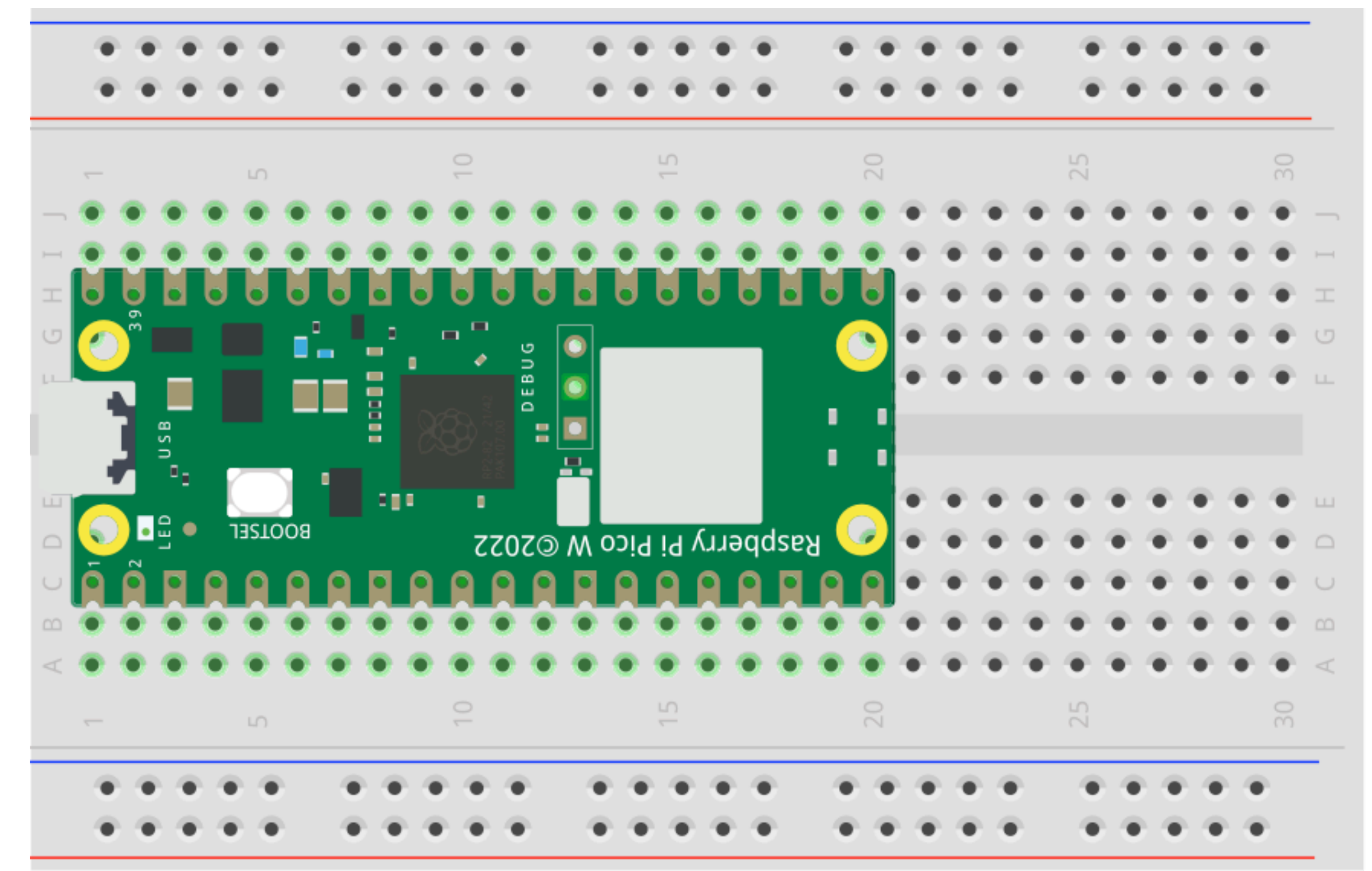
Temperatur mit dem integrierten Temperatursensor messen (1)

Auf der Platine des Raspberry Pi Pico ist ein Temperatursensor drauf, der sich für Temperaturmessungen verwenden lässt. Die Temperatur lässt sich quasi messen, auslesen, auswerten und für die Steuerung und Regelung nutzen.

Die Temperatur-Messung mit dem im Pico eingebauten Temperatur-Sensor ist ein gutes Beispiel für die Komplexität und Probleme mit der Hardware-nahen Programmierung. Die eigentliche Temperaturmessung ist die Messung einer Spannung an einer Halbleiterdiode. Da die Leitfähigkeit von Halbleitern temperaturabhängig ist, eignen sie sich als Temperatursensor. Die Messung der Temperatur findet also über einen Umweg statt.

Das Umrechnen in eine Temperatur muss man also programmieren.

Da der Raspberry Pi Pico keine Anzeige hat, findet die Temperatur-Anzeige im Editor-Feld „Kommandozeile“ bzw. „Shell“ der Thonny Python IDE statt.



Temperatur mit dem integrierten Temperatursensor messen (2)

Nach dem Start des Programms wird die Temperatur ermittelt. Die dazugehörigen Werte, „Dezimalzahl“, „Spannung“ und „Temperatur“ werden angezeigt.

Wichtig zu verstehen ist, dass alle drei Werte das gleiche bedeuten. Sie unterscheiden sich nur in der Darstellung.

Wie lässt sich die Temperatur erhöhen und senken?

- Eine höhere Temperatur erreichst Du dadurch, dass Du den Pico anhauchst. Die Luft aus unserem Atem hat in der Regel eine höhere Temperatur als die Umgebung.
- Um die Temperatur zu senken, kannst Du einfach warten und zuschauen, wie Temperatur fällt. Oder Du pustest die warme Luft am Pico weg.

```
# Bibliotheken laden
from machine import ADC
from time import sleep

# Initialisierung des ADC4
sensor_temp = ADC(4)
conversion_factor = 3.3 / (65535)

# Wiederholung einleiten (Schleife)
while True:
    # Temperatur-Sensor als Dezimalzahl lesen
    read = sensor_temp.read_u16()
    # Dezimalzahl in eine reelle Zahl umrechnen
    spannung = read * conversion_factor
    # Spannung in Temperatur umrechnen
    temperatur = 27 - (spannung - 0.706) / 0.001721
    # Ausgabe in der Kommandozeile/Shell
    print("Dezimalzahl: ", read)
    print("Spannung (V): ", spannung)
    print("Temperatur (°C): ", temperatur)
    print()
    # 2 Sekunden warten
    sleep(3)
```

Raspberry Pi Pico W als WLAN-Client (1)

Der Raspberry Pi Pico W hat auf seiner Platine einen Funk-Chip mit WLAN-Unterstützung. Da ist es natürlich naheliegend den Pico mit einem vorhandenen WLAN-Netzwerk zu verbinden. Alles was man braucht ist den Namen (SSID) des betreffenden WLANs und das zugehörige Passwort zur Authentifizierung.

Zur Herstellen der WLAN-Verbindung reichen im einfachsten Fall schon insgesamt nur 4 Zeilen Programmcode aus.

1. Die erste Zeile lädt die Bibliothek für die Netzwerk-Funktionen.
2. Hier wird die Betriebsart definiert. STA_IF bedeutet so viel wie Station, also der Client-Betrieb.
3. Jetzt wird das WLAN-Interface aktiviert.
4. Die Verbindung wird mit dem WLAN-Namen (SSID) und dem Passwort (PASSWORD) hergestellt.

Die nachfolgenden 4 Zeilen haben mit dem Verbindungsaufbau nichts zu tun. Sie sind eine sinnvolle Ergänzung. Hier wird geprüft, ob die Verbindung erfolgreich war.

```
# Bibliotheken laden (1)
import network

# Ländereinstellung
network.country('DE')

# Hostname einstellen
network.hostname("PicoW")

# Client-Betrieb (2)
wlan = network.WLAN(network.STA_IF)

# WLAN-Interface aktivieren (3)
wlan.active(True)

# WLAN-Verbindung herstellen (4)
wlan.connect('SSID', 'PASSWORD')

# WLAN-Verbindungsstatus prüfen
import time
print('Warten auf WLAN-Verbindung')
while not wlan.isconnected() and wlan.status() >= 0:
    time.sleep(1)
print('WLAN-Verbindung hergestellt / Status:', wlan.status())
```

Hinweis: Nach der Meldung „WLAN-Verbindung hergestellt“ wird das Programm beendet. Die WLAN-Verbindung des Picos bleibt dabei bestehen. Der WLAN-Chip auf dem Pico-Board hält die WLAN-Verbindung unabhängig welches Programm auf dem Pico ausgeführt wird.

Raspberry Pi Pico W als WLAN-Client (2)

Was macht der folgende Programmcode besser?

1. Das Herstellen der WLAN-Verbindung befindet sich in einer Funktion. Die Funktion kann in einen eigenen Programmcode übernommen werden und mit einem Kommando aufgerufen werden.
2. Die Funktion prüft, ob schon eine WLAN-Verbindung besteht.
3. Ausgabe zusätzlicher Informationen auf der Kommandozeile über die Netzwerk-Verbindung (IPv4-Konfiguration).
4. Onboard-LED als Status-Anzeige: Wenn sie blinkt, dann wird die WLAN-Verbindung hergestellt. Leuchtet die LED, dann besteht die Verbindung zum WLAN.

```
# Bibliotheken laden
import machine
import network
import time

# WLAN-Konfiguration
wlanSSID = 'SSID'
wlanPW = 'PASSWORD'
network.country('DE')

# Status-LED
led_onboard = machine.Pin('LED', machine.Pin.OUT, value=0)

# Funktion: WLAN-Verbindung (1)
def wlanConnect():
    wlan = network.WLAN(network.STA_IF)
    if not wlan.isconnected(): # (2)
        print('WLAN-Verbindung herstellen')
        wlan.active(True)
        wlan.connect(wlanSSID, wlanPW)
        for i in range(10):
            if wlan.status() < 0 or wlan.status() >= 3:
                break
            led_onboard.toggle()
            print('.', wlan.status())
            time.sleep(1)
    if wlan.isconnected():
        print('WLAN-Verbindung hergestellt')
        led_onboard.on() # (4)
        print('WLAN-Status:', wlan.status())
        netConfig = wlan.ifconfig() # (3)
        print('IPv4-Adresse:', netConfig[0], '/', netConfig[1])
        print('Standard-Gateway:', netConfig[2])
        print('DNS-Server:', netConfig[3])
    else:
        print('Keine WLAN-Verbindung')
        led_onboard.off() # (4)
        print('WLAN-Status:', wlan.status())

# WLAN-Verbindung herstellen
wlanConnect()
```


Statische IPv4-Konfiguration

Für eine funktionierende Netzwerk-Verbindung benötigt der Pico noch eine vollständige IPv4-Konfiguration. Die bekommt er vom DHCP-Server im selben Netzwerk halbautomatisch zugeteilt. Das bedeutet aber auch, dass die dazugehörige IPv4-Adresse dynamisch zugeteilt wird. Also im Prinzip unbekannt ist und sich wegen der dynamischen Zuteilung durch DHCP auch mal ändern kann.

Wenn Netzwerk-Teilnehmer dauerhaft unter derselben IPv4-Adresse erreichbar sein sollen, dann muss die IPv4-Konfiguration auf dem Pico im Programmcode manuell vorgenommen werden. Das ist mit einer Zeile Programmcode erledigt. Doch die 4 notwendigen Parameter müssen gekannt sein oder müssen ermittelt werden. Ein bisschen Ahnung von IPv4 und Netzwerk-Konfiguration solltest Du mitbringen.

Aus was besteht eine vollständige IPv4-Konfiguration?

- IPv4-Adresse für den Host (WLAN-Client)
- Subnetzmaske
- IPv4-Adresse des Standard-Gateways im Netzwerk
- IPv4-Adresse des DNS-Servers im Netzwerk

```
# Bibliotheken laden
import network

# Ländereinstellung
network.country('DE')

# Client-Betrieb
wlan = network.WLAN(network.STA_IF)

# WLAN-Interface aktivieren
wlan.active(True)

# IPv4-Konfiguration
# ('IPv4-Adresse', 'Subnetzmaske', 'Standard-Gateway', 'DNS-Server')
wlan.ifconfig(('192.168.178.2', '255.255.255.0', '192.168.178.1', '192.168.178.1'))

# WLAN-Verbindung herstellen
wlan.connect('SSID', 'PASSWORD')
```

Hinweis: Die im Programmcode verwendete IPv4-Konfiguration ist ein Beispiel. Es muss in Ihrem Netzwerk nicht funktionieren.

Internet-Verbindung prüfen

Wenn man den Raspberry Pi Pico W als WLAN-Client programmiert und eine Verbindung zum Internet braucht, dann will man vielleicht prüfen, ob eine Verbindung ins Internet über eine bestehende WLAN-Verbindung möglich ist.

Nur weil eine WLAN-Verbindung erfolgreich war, bedeutet das nicht, dass man damit auch Zugang zum Internet hat oder eine Internet-Verbindung möglich ist. Deshalb wollen wir hier mit dem folgenden Programmcode eine Internet-Verbindung prüfen.

Der folgende Programmcode ist ohne Netzwerk-Verbindung nicht lauffähig. Da der Funk-Chip vom Mikrocontroller-Chip unabhängig ist, reicht es aber aus, vor diesem Programmcode einen Programmcode aufzurufen, der eine Verbindung zum WLAN herstellt.

Siehe Raspberry Pi Pico W als WLAN-Client (1) oder (2).

```
# Bibliotheken laden
import requests

try:
    # HTTP-Request senden
    request = requests.get('http://www.das-elko.de/')
    # HTTP-Response-Status-Code ausgeben
    print('Status-Code:', request.status_code)
    # Verbindung schließen
    request.close()
except OSError:
    print('Fehler: Keine Netzwerk-Verbindung (WLAN)')
```

Hinweis: Ein Status-Code (HTTP-Response-Code) von „200“ bedeutet, dass die Verbindung zu dem betreffenden Server erfolgreich war.

Bei anderen Status-Codes wurde der Server erreicht, aber der Request hat einen Fehler auf dem Server verursacht. Die Internet-Verbindung war dann trotzdem erfolgreich.

Datum und Uhrzeit per NTP-Zeitserver einstellen

Der Raspberry Pi Pico hat eine Echtzeituhr (Real-time Clock, RTC), die aber ohne eigene Stromversorgung die Uhrzeit nicht mitführen kann. Das heißt, die interne RTC beginnt bei jedem Neustart wieder von vorne. Es sind zwar zeitabhängige Funktionen möglich, aber nicht bezogen auf eine bestimmte Uhrzeit oder ein Datum.

Dank der Verbindung zum Internet über WLAN ist es jedoch möglich, das aktuelle Datum und die Uhrzeit von einem Zeitserver abzurufen. Hierfür gibt es mit NTP (Network Time Protocol) sogar ein eigenes Kommunikationsprotokoll.

Es geht also darum, Datum und Uhrzeit der internen RTC beim Neustart richtig einzustellen. Dadurch sind Datum- oder Uhrzeit-abhängige Funktionen möglich.

Hinweis: Beachte hierzu die Hinweise zur Zeitzone und Zeitumstellung.

Aufbau, Beschreibung und Programmcode

<https://www.elektronik-kompendium.de/sites/raspberry-pi/2708151.htm>

Betrieb als Webserver

Mit seiner WLAN-Unterstützung ist ein Raspberry Pi Pico W nicht nur als Client, sondern auch als Webserver geeignet. Ein Webserver kann im Netzwerk auf Anfragen Daten zurücksenden. Und dabei hat ein Raspberry Pi Pico W einiges an Daten zu bieten. Beispielsweise die Zustände von GPIO-Eingängen und -Ausgängen, sowie digitale Werte an den ADC-Eingängen.

Hinweis: Die Implementierung eines vollständigen Webserver macht auf einem Raspberry Pi Pico W keinen Sinn. Hierfür würde sich ein Mini-Computer, wie der Raspberry Pi 3/4/Zero besser eignen.

Aufbau, Beschreibung und Programmcode

<https://www.elektronik-kompodium.de/sites/raspberry-pi/2707131.htm>

E-Mail senden

Die Idee ist, dass ein Raspberry Pi Pico W eine E-Mail an seinen Nutzer sendet, wenn ein Ereignis ausgelöst wird oder Daten gesendet werden müssen. Wenn man von einem Pico nicht ständig Daten anfordern will, ist das eine praktische Lösung. Du kannst Dir Informationen und Daten vom Pico direkt in Dein E-Mail-Postfach liefern lassen.

Aufbau, Beschreibung und Programmcode

<https://www.elektronik-kompodium.de/sites/raspberry-pi/2710141.htm>

Online-Wetterstation mit Vorhersage

Eine Wetterstation ist in der Regel eine Zusammenstellung verschiedener Messgeräte, die zur Messung meteorologischer Größen dienen. In diesem Fall ist es ein Raspberry Pi Pico W, der verschiedene meteorologische Größen nicht lokal, sondern aus einer API bezieht, die vom Deutschen Wetterdienst (DWD) kostenlos bereitgestellt wird. Auf diese Weise entsteht eine Online-Wetterstation mit Vorhersage für einen Ort (auf der ganzen Welt), an dem eine Wetterstation betrieben wird, von der der DWD meteorologische Daten erhält.

Hinweis: Eine API (Application Programming Interface) ist vereinfacht gesagt eine Programmierschnittstelle eines entfernten oder fremden Systems mit dem man Daten austauschen kann. Im einfachsten Fall stellt eine API Daten zum Abruf bereit. In der Regel kann man über bestimmte Parameter den Datenumfang genauer spezifizieren oder eingrenzen. Manche APIs haben auch die Möglichkeit Daten entgegenzunehmen.

Die meisten APIs erfordern es, dass sich der Nutzer vorher anmeldet und die Software, die auf die API zugreift bei jedem Zugriff authentifiziert (mit einem API-Token). Nur selten sind APIs öffentlich, ohne Authentifizierung, zugänglich.

Aufbau, Beschreibung und Programmcode

<https://www.elektronik-kompendium.de/sites/raspberry-pi/2709171.htm>

MQTT-Publisher und MQTT-Subscriber

Die MQTT-Architektur kennt 3 Rollen: Publisher, Subscriber und Broker. Mit der entsprechenden Programmierung kann ein Raspberry Pi Pico W die Rolle des MQTT-Publishers oder eines -Subscribers einnehmen. Einen Pico als Broker einzusetzen macht eigentlich keinen Sinn. Dafür würde sich ein Mini-Computer, wie der Raspberry Pi 3/4/Zero besser eignen.

MQTT-Publisher: Schnurloser Taster

<https://www.elektronik-kompendium.de/sites/raspberry-pi/2709091.htm>

MQTT-Publisher: Schnurloses Thermometer

<https://www.elektronik-kompendium.de/sites/raspberry-pi/2709101.htm>

MQTT-Subscriber: Schnurloser Schalter

<https://www.elektronik-kompendium.de/sites/raspberry-pi/2709111.htm>

Online-Workshop: Picobello

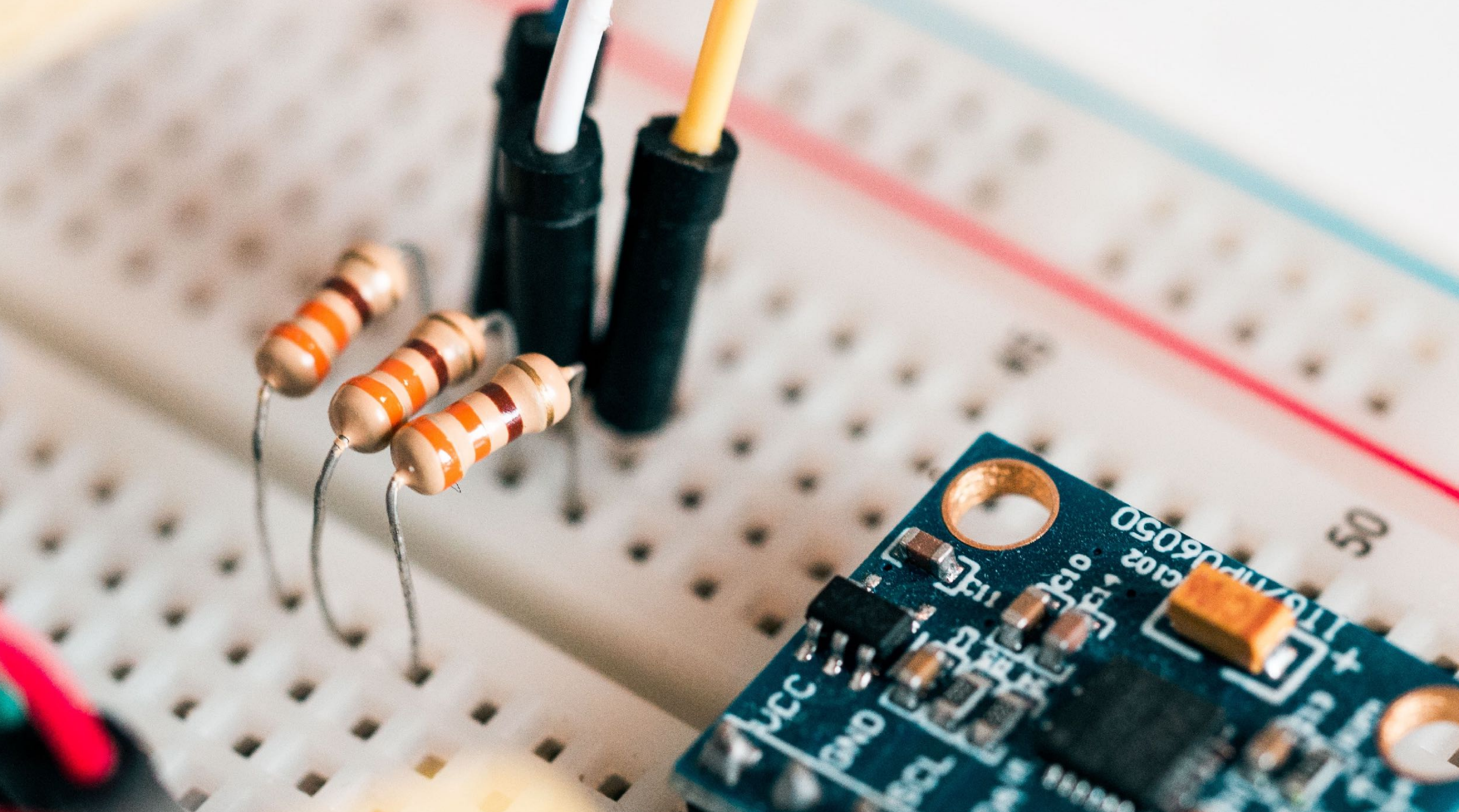
Du musst das
nicht alleine
machen!



Im Online-Workshop PicoBello ...

- ... Lernst Du Programmieren mit dem Raspberry Pi Pico.
- ... Wir arbeiten mit den Teilnehmern gemeinsam.
- ... Bekommst Du Unterstützung bei individuellen Problemen.

<https://www.elektronik-kompendium.de/service/events/>



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

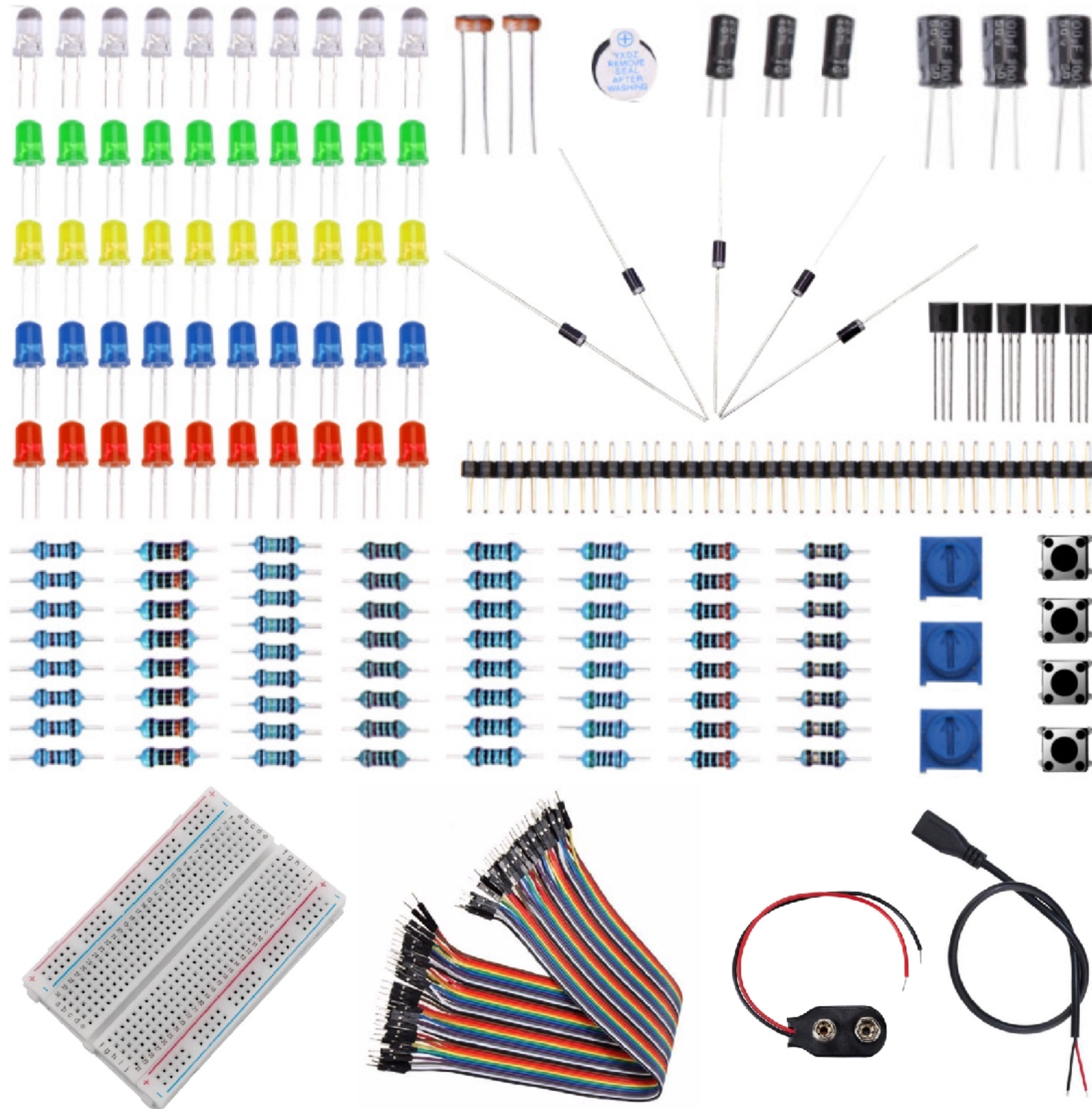


Das Elektronik-Set Eingabe Ausgabe Edition ist eine Sammlung beliebter Bauteile zur Dateneingabe, Steuerung und Datenausgabe für die Hardware-nahe Programmierung mit Mikrocontrollern und Mini-Computern.

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/ingabe-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 - 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-kompodium.de/shop/buecher/elektronik-fibel>