EXPLORER SET

RB-P-XPLR-SET

INHALTSVERZEICHNIS

1. ALLGEMEINE INFORMATIONEN

Sehr geehrte*r Kunde*in,
vielen Dank, dass Sie sich für unser Produkt entschieden haben. Im Folgenden zeigen wir Ihnen, was bei der Inbetriebnahme und der Verwendung zu beachten ist.

Sollten Sie während der Verwendung unerwartet auf Probleme stoßen, so können Sie uns selbstverständlich gerne kontaktieren.

2. GERÄTEÜBERSICHT & PIN-BELEGUNG

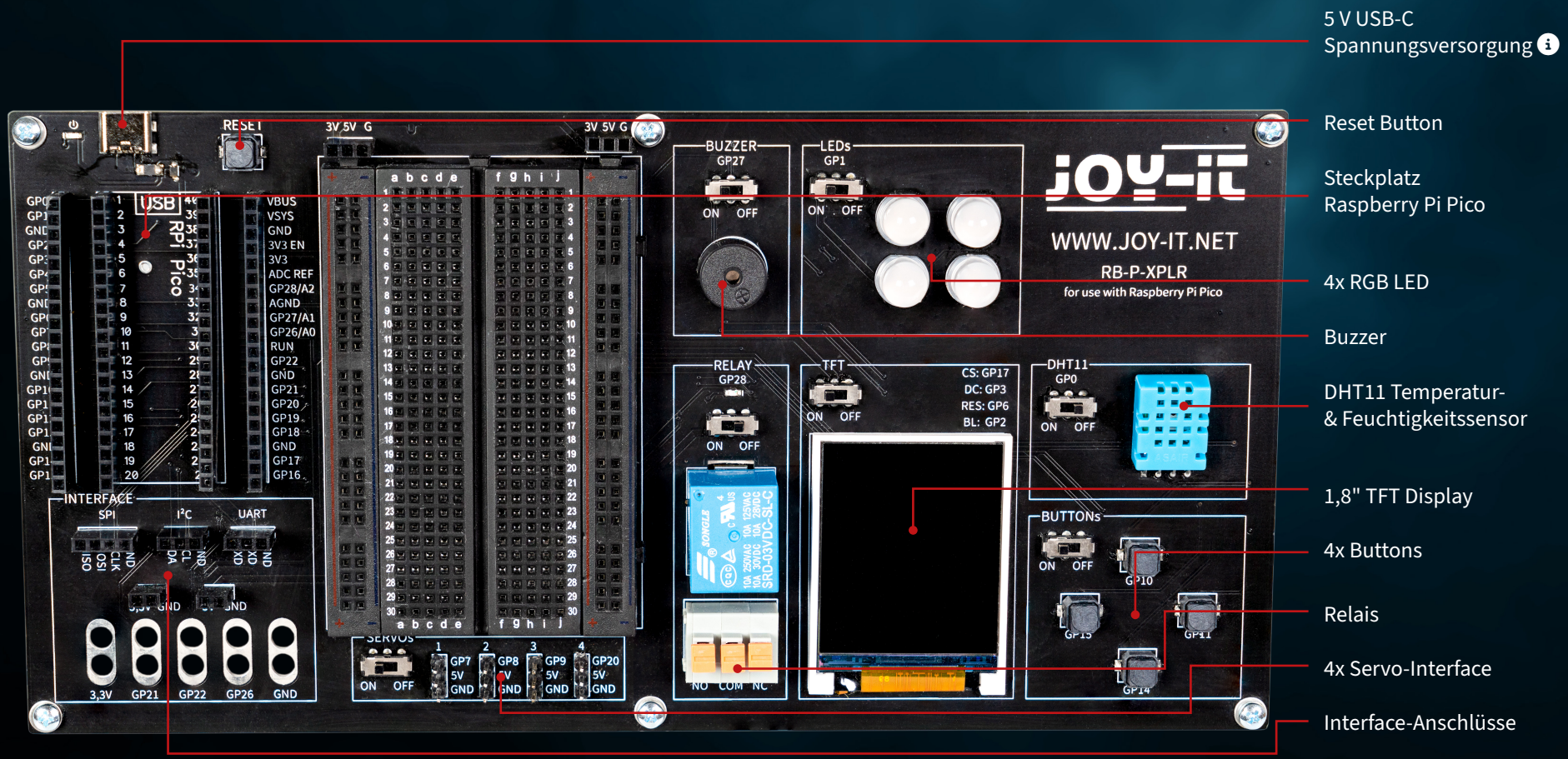
Unser Explorer Board ist die einfache und effiziente Möglichkeit, Ihre Raspberry Pi Pico Projekte zu entwickeln.

Da die wichtigsten Komponenten bereits integriert sind, sparen Sie Zeit und Mühe beim Verkabeln. Das Explorer Board verfügt über eine breite Palette an Interface-Anschlüssen, sodass Sie Ihre Projekte mit einer Vielzahl von Modulen und Geräten verbinden können. Mit dem integrierten Breadboard lassen sich eigene Projekte schnell aufbauen und realisieren.

Dank der Möglichkeit, alle Module einzeln zu- oder abzuschalten, können Sie Ihre Pins, welche zusätzlich separat nach außen geführt sind, jederzeit für andere Projekte nutzen oder auf dem integrierten Breadboard experimentieren.

Alle verbauten Komponenten lassen sich über den jeweiligen Schalter ausschalten, falls diese nicht benötigt werden. So können die zugehörigen Pins, falls nötig, auch für andere Komponenten verwendet werden.

Links und Rechts des Raspberry Pi Picos sind alle Pins noch einmal zusätzlich ausgeführt. Hier können Komponenten direkt angeschlossen oder über zusätzliche Kabel zum integrierten Breadboard geführt werden.

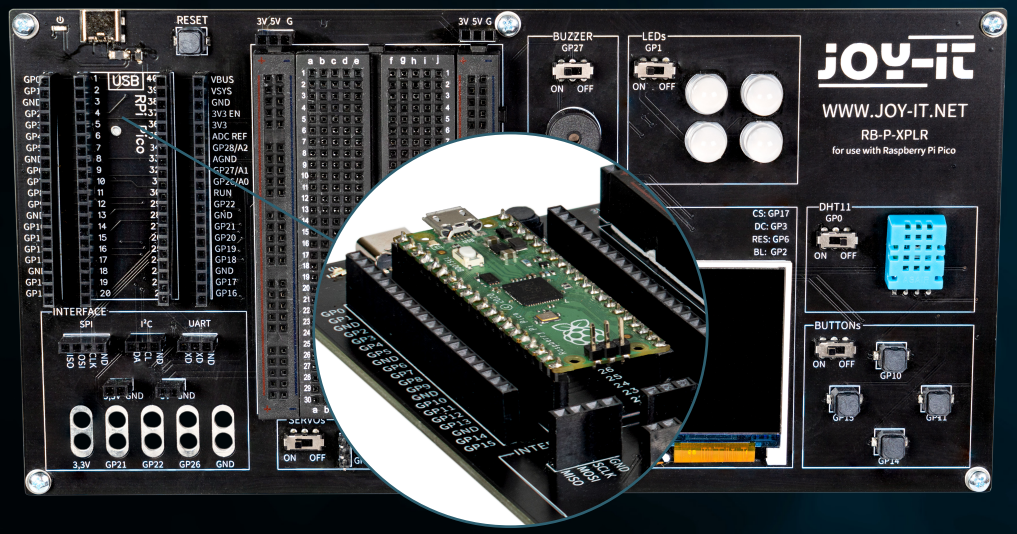


Bitte beachten Sie, dass zur Verwendung stets die USB-C Verbindung angeschlossen sein muss. Eine Stromversorgung über den Micro-USB Anschluss des Raspberry Pi Picos ist nicht möglich.

PIN-BELEGUNG	
Buzzer	GP27
LEDs	GP1
Relais	GP28
1,8" TFT Display	CS: GP17, DC: GP3, RES: GP6, BL: GP2
DHT11	GP0
Buttons	GP10, GP11, GP14 & GP15
Servos	GP7, GP8, GP9 & GP20
UART	RXD: GP13, TXD: GP12
I2C	SDA: GP4, SCL: GP5
SPI	MISO: GP16, MOSI: GP19, SCLK: GP18

3. RASPBERRY PI PICO

Stecken Sie zunächst Ihen Raspberry Pi Pico auf dem Steckplatz Ihres Boards ein.



Schließen Sie zur Programmierung nun ein Micro-USB Kabel an Ihren Computer und an den Raspberry Pi Pico an.

ACHTUNG! Der USB-C Anschluss auf dem Explorer-Board dient ausschließlich zur Stromversorgung. Hierüber werden keine Daten an den Raspberry Pi übertragen.

Zur Übertragung unseres Beispielprogramms können Sie ein geeignetes Entwicklungsprogramm Ihrer Wahl verwenden. Wir empfehlen hier die **Thonny Python IDE**.

ACHTUNG! Wenn Sie neu in der Welt der Mikrocontroller und Elektronik sind, keine Sorge! Wir haben eine spezielle Anfängeranleitung für Sie vorbereitet. Diese Anleitung ist speziell auf die Bedürfnisse von Anfängern zugeschnitten und erklärt Ihnen die Verwendung des Raspberry Pi Picos Schritt für Schritt.

Von der Grundkonfiguration bis hin zur Ausführung von Projekten - in dieser Anleitung begleiten wir Sie durch den gesamten Prozess. Unsere Anleitung umfasst leicht verständliche Erklärungen und nützliche Tipps um Ihnen zu helfen, Ihre Fähigkeiten im Umfang mit dem Raspberry Pi Pico schnell und effektiv zu entwickeln. Unsere Anleitung können Sie **hier** herunterladen.

4. MODULE IM DETAIL

Im Folgenden werden alle Module, die auf dem Explorer Board vorhanden sind, einzeln mit Beispielcodes erläutert. **Hier** können Sie alle Beispielcodes und Bibliotheken herunterladen, sowie einen Beispielcode, der alle Module miteinander verbindet.

Für die Verwendung einiger Module werden externe Bibliotheken, sowie eine Schrift-Datei verwendet. Laden Sie sich die Bibliotheken herunter und laden Sie diese in den lib-Ordner Ihres Raspberry Pi Picos. Platzieren Sie die Schrift-Datei im Hauptverzeichnis Ihres Raspberry Pi Picos.

4.1 BUZZER

Ein Buzzer oder Summer erzeugt einen Signalton, ähnlich wie ein Lautsprecher. Im Gegensatz zum Lautsprecher eignet er sich jedoch nur für einen eingeschränkten Frequenzbereich, sodass er keinen guten Klang zur Wiedergabe von Musik oder Sprache erzeugt. Dafür eignet er sich hervorragend, um laute Warntöne in Form von Piepsern zu erzeugen. Wann immer ein elektrisches Gerät einen Warnton erzeugt, übernimmt das fast immer ein Buzzer. Zum Beispiel im Wecker, Rauchmelder oder der Gurtwarner im Auto.

Der Buzzer ist an den GPIO Pin GP27 angeschlossen.

```
# Load libraries
from machine import Pin, PWM

buzzerPin = Pin(27)
buzzer = PWM(buzzerPin)

while True:
    # Activate buzzer for 1 sec
    buzzer.freq(1000)
    buzzer.duty_u16(1000)
    sleep(1)
    buzzer.duty_u16(0)
    sleep(1)
```

4.2 RGB LEDS

RGB-LEDs sind eine Art Leuchtdiode, die Rot, Grün und Blau kombiniert, um eine Vielzahl von Farben zu erzeugen. Ähnlich wie ein Buzzer nur einfache Töne erzeugt, können RGB-LEDs keine komplexen Bilder darstellen, aber sie sind hervorragend darin, Farben zu mischen und zu variieren. Jede LED in einer RGB-Einheit kann in ihrer Intensität variiert werden, um unterschiedliche Farbtöne zu erzeugen, von sanften Pastellfarben bis hin zu leuchtenden, satten Farben. Dies macht sie ideal für Stimmungsbeleuchtung, dekorative Beleuchtung und in Anwendungen, wo visuelle Signale erforderlich sind, wie in Gaming-Setups oder als Statusanzeigen in elektronischen Geräten. Ihre Vielseitigkeit und Energieeffizienz haben sie zu einer beliebten Wahl in modernen Beleuchtungssystemen gemacht, obwohl sie, ähnlich wie der Buzzer, aufgrund ihrer einfachen Funktionsweise keine komplexen Bilder oder Muster ohne zusätzliche Steuerungseinheiten erzeugen können.

Der GPIO LEDs sind an den GPIO Pin GP1 angeschlossen.

```
# Load libraries
from machine import Pin, PWM
from utime import sleep
from neopixel import NeoPixel

ledPin = 1
ledCount = 4

# Initialize GPIOs
led = Pin(ledPin, Pin.OUT)
led = NeoPixel(Pin(ledPin, Pin.OUT), ledCount)

while True:
    # Turn LEDs white
    for i in range (ledCount):
        led[i] = (255, 255, 255)
    led.write()
    sleep(1)
    # Turn LEDs red
    for i in range (ledCount):
        led[i] = (255, 0, 0)
    led.write()
    sleep(1)
    # Turn LEDs blue
    for i in range (ledCount):
        led[i] = (0, 0, 255)
    led.write()
    sleep(1)
    # Turn LEDs green
    for i in range (ledCount):
        led[i] = (0, 255, 0)
    led.write()
    sleep(1)
```

4.3 RELAIS

Relais sind einige der ältesten elektromechanischen Komponenten und funktionieren als elektrisch gesteuerte Schalter. Mit einer kleinen Eingangsspannung und geringem Strom kann eine große elektrische Last am Ausgang ein- und ausgeschaltet werden. Wenn das Relais durchschaltet, leuchtet auch die rote LED. In die Klemmbuchse können Sie abisolierte Kabelenden einstecken (durch Niederdrücken des orangen Hebels), um die drei Anschlüsse zu nutzen.

Das Relais ist an den GPIO Pin GP28 angeschlossen.

```
# Load libraries
from machine import Pin, PWM
from utime import sleep

relayPin = 28
# Initialize GPIOs
relay = Pin(relayPin, Pin.OUT)

while True:
    # Toggle Relay
    relay.on()
    sleep(1)
    relay.off()
    sleep(1)
```

4.4 TFT

Das Flüssigkristalldisplay (LCD TFT) mit rund 65.000 Farben und einer Diagonale von 1,8 Zoll hat eine Auflösung von 128×160 Pixeln und kann über SPI angesteuert werden. Es eignet sich für die Darstellung von bunten Grafiken und Bildern. Buchstaben und andere Zeichen werden dabei als Grafiken dargestellt, die aus vielen einzelnen Punkten zusammengesetzt sind.

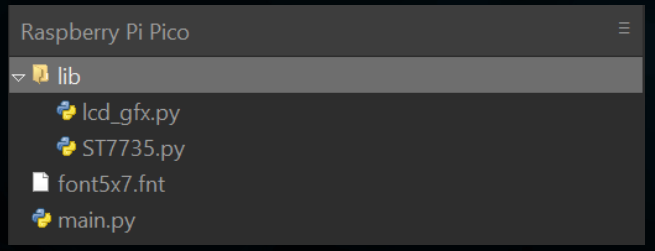
Das TFT ist an die GPIO Pins GP17 (CS), GP3 (DC), GP6 (RES) und GP2 (BL) angeschlossen.

```
from machine import Pin, SPI
import ST7735

# Initialize LCD
spi = SPI(0, baudrate=8000000, polarity=0, phase=0, sck=Pin(18), mosi=Pin(19),
miso=Pin(16))
lcd = ST7735.ST7735(spi, rst=6, ce=17, dc=3)
backlight = Pin(2, Pin.OUT)

# Turn backlight on
backlight.high()
lcd.reset()
lcd.begin()

# Display content on the LCD
lcd.fill_screen(lcd.rgb_to_565(0, 255, 0))  # Fills the screen with a green color

# Display text
lcd.p_string(20, 50, ‚Hello, World!‘)
```

Neben Texten können beispielsweise auch Rechtecke dargestellt werden:

```
# Draw red rectangle
lcd.draw_block(10, 10, 50, 50, lcd.rgb_to_565(255, 0, 0))
```

ACHTUNG! Für das TFT-Display werden zwei separate Bibliotheksdateien, sowie eine Schriftart-Datei benötigt.Sie können sich die benötigten Dateien **hier** herunterladen. Übertragen Sie dann alle Dateien aus dem Libraries-Ordner in das Hauptverzeichnis Ihres Raspberry Pi Picos, sodass die Ordnerstruktur dann wie folgt aussieht:



4.5 DHT 11

Der DHT11-Sensor kann Temperaturen von 0 °C bis 50 °C (±2 °C Genauigkeit) und eine relative Luftfeuchtigkeit von 20 % bis 80 % (±5 %) erfassen (höchstens einmal pro Sekunde). Wetterstationen sind vermutlich das primäre Einsatzgebiet für einen Sensor wie den DHT11. Um die Funktionalität zu testen, genügt es, den Mund nahe über den Sensor zu halten und langsam auszuatmen. Die Atemluft unterscheidet sich hinsichtlich Temperatur und Feuchtigkeit von der Umgebung, was zu einer deutlichen Änderung der Werte führen sollte.

Der DHT11 ist an den GPIO Pin GP0 angeschlossen.

```
from machine import Pin
from dht import DHT11
from utime import sleep

# Initialize DHT11 Sensor
dhtPin = 0
dht = DHT11(Pin(dhtPin, Pin.IN))

while True:
    # Measure DHT11 values
    dht.measure()
    temp = dht.temperature()  # Temperature in Celsius
    humid = dht.humidity()    # Relative Humidity in %

    # Print the measurements
    print('Temperature:', temp, '°C')
    print('Humidity:', humid, '%')

    sleep(2)  # Wait for 2 seconds before the next reading
```

4.6 BUTTONS

Buttons sind interaktive Elemente in Benutzeroberflächen, die eine einfache, aber essenzielle Funktion erfüllen: die Benutzereingabe. Ähnlich wie RGB-LEDs eine Vielzahl von Farben anzeigen können, dienen Buttons dazu, eine breite Palette von Befehlen und Aktionen in digitalen Umgebungen zu initiieren.

Die Buttons sind an die GPIO Pins GP10 (Oben), GP11 (Rechts), GP14 (Unten) und GP15 (Links) angeschlossen.

```
from machine import Pin

# Define button pins
buttons = [10, 11, 14, 15]

# Initialize buttons
buttonOne = Pin(buttons[0], Pin.IN, Pin.PULL_DOWN)
buttonTwo = Pin(buttons[1], Pin.IN, Pin.PULL_DOWN)
buttonThree = Pin(buttons[2], Pin.IN, Pin.PULL_DOWN)
buttonFour = Pin(buttons[3], Pin.IN, Pin.PULL_DOWN)

# Define button handler functions
def buttonUp(pin):
    print("Button Up Pressed")

def buttonRight(pin):
    print("Button Right Pressed")

def buttonDown(pin):
    print("Button Down Pressed")

def buttonLeft(pin):
    print("Button Left Pressed")

# Attach interrupt handlers to buttons
buttonOne.irq(trigger=Pin.IRQ_RISING, handler=buttonUp)
buttonTwo.irq(trigger=Pin.IRQ_RISING, handler=buttonRight)
buttonThree.irq(trigger=Pin.IRQ_RISING, handler=buttonDown)
buttonFour.irq(trigger=Pin.IRQ_RISING, handler=buttonLeft)
```

4.7 SERVOS

Ein Servo besteht aus einem Elektromotor mit Getriebe und Steuerelektronik. Auf der Ausgangsseite des Getriebes befindet sich ein Zahnrad, auf das das Servohorn montiert wird. Der Servo kann die Achse in einem Bereich von etwa 180° bewegen. Einsatzgebiete für Servos sind zum einen der Modellbau, um bei einem Flugzeug oder Schiff beispielsweise die Flügel- oder Ruderstellung zu kontrollieren. Auch im Automobilbau werden immer mehr Servos verbaut, um Türen automatisch zu schließen, für Fensterheber, Spiegel und andere verstellbare Elemente.

Die Servo-Anschlüsse sind die GPIO Pins GP7, GP8, GP9 und GP20.

```
from machine import Pin, PWM
from utime import sleep

# Servo pin numbers
servoOnePin = 7
servoTwoPin = 8
servoThreePin = 9
servoFourPin = 20

# Initialize servos
servoOne = PWM(Pin(servoOnePin))
servoTwo = PWM(Pin(servoTwoPin))
servoThree = PWM(Pin(servoThreePin))
servoFour = PWM(Pin(servoFourPin))

# Servo degree positions in nanoseconds
deg0 = 500000
deg45 = 1000000
deg90 = 1500000
deg135 = 2000000
deg180 = 2500000

while True:
    # Move each servo through a range of angles
    for servo in [servoOne, servoTwo, servoThree, servoFour]:
        servo.duty_ns(deg0)
        sleep(1)
        servo.duty_ns(deg45)
        sleep(1)
        servo.duty_ns(deg90)
        sleep(1)
        servo.duty_ns(deg135)
        sleep(1)
        servo.duty_ns(deg180)
        sleep(1)
```

4.8 INTERFACES

Interface-Anschlüsse spielen eine entscheidende Rolle in der Welt der Elektronik, ähnlich wie Buttons in Benutzeroberflächen. Sie ermöglichen die Kommunikation und Stromversorgung zwischen verschiedenen elektronischen Bauteilen. Auf unserem Explorer Board sind daher im Interface-Bereich die folgenden Anschlüsse zu finden:

SPI (Serial Perhipheral Interface): Dieser Anschluss wird für die schnelle serielle Datenübertragung verwendet. Er besteht typischerweise aus vier Leitungen: MISO (Master In, Slave Out), MOSI (Master Out, Slave In), SCK (Serial Clock) und SS (Slave Select). SPI eignet sich hervorragend für Situationen, in denen eine hohe Datenübertragungsrate erforderlich ist, wie bei der Ansteuerung von LCD-Displays oder SD-Karten.

I2C (Inter-Integrated Circuit): I2C ist ein Zwei-Draht-Interface, bestehend aus einer Datenleitung (SDA) und einer Taktleitung (SCL). Es wird häufig in Mikrocontroller-Anwendungen für die Kommunikation zwischen verschiedenen integrierten Schaltungen verwendet. Seine Einfachheit macht es ideal für Anwendungen, in denen nicht viele GPIO-Pins zur Verfügung stehen.

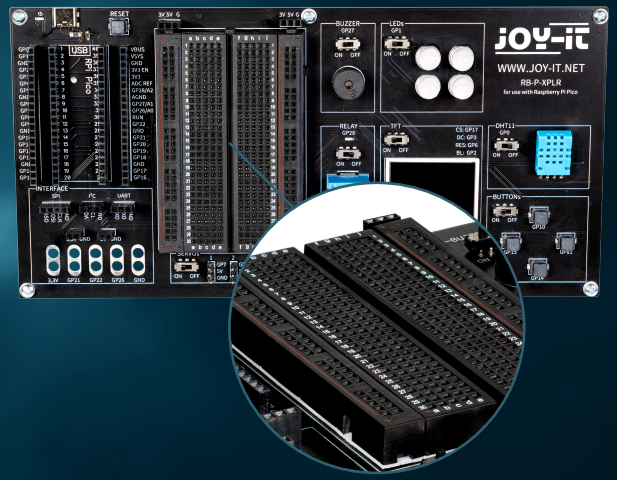
UART (Universal Asynchronous Receiver/Transmitter): Dieses Interface ermöglicht eine asynchrone serielle Kommunikation über zwei Leitungen: TX (Transmit) und RX (Receive). UART wird häufig für die Kommunikation zwischen Mikrocontrollern und Computern oder für die Anbindung von Modulen wie GPS-Empfängern oder Bluetooth-Modulen genutzt.

3,3 V und 5 V Anschlüsse: Diese Anschlüsse stellen die Stromversorgung für elektronische Komponenten bereit. 3,3 V wird oft für moderne Mikrocontroller und Sensoren verwendet, während 5 V häufig bei älteren oder leistungshungrigeren Geräten anzutreffen ist.

Anschlüsse für Krokodilklemmen: Diese Anschlüsse sind ideal für temporäre Verbindungen oder für Testzwecke. Sie ermöglichen eine einfache und schnelle Verbindung zu verschiedenen Komponenten oder Messgeräten ohne Löten. Insgesamt gibt es auf dem Explorer Board fünf solcher Anschlüsse, die für eine Vielzahl von Anwendungen flexibel eingesetzt werden können.

Jeder dieser Anschlüsse hat seine spezifische Anwendung und Bedeutung in der Elektronik, ähnlich wie verschiedene Arten von Buttons in einer Benutzeroberfläche unterschiedliche Funktionen haben. Sie bieten die notwendige Flexibilität und Funktionalität für den Aufbau und die Erweiterung elektronischer Systeme.

4.9 BREADBOARD



Breadboards sind ein unverzichtbares Werkzeug in der Welt der Elektronik, ähnlich wie Interface-Anschlüsse für die Verbindung verschiedener Komponenten entscheidend sind. Sie ermöglichen es, elektronische Schaltungen schnell und ohne Löten aufzubauen und zu testen, was sie besonders bei Prototyping und Bildungszwecken beliebt macht.

Ein Breadboard besteht typischerweise aus einem rechteckigen Kunststoffblock mit einer Vielzahl von eingebetteten Löchern, die in Reihen angeordnet sind. Diese Löcher sind intern durch metallische Leiterbahnen verbunden, die es ermöglichen, Komponenten und Drähte einfach einzustecken und zu verbinden. Die Standardanordnung eines Breadboards umfasst zwei Hauptbereiche:

Die Hauptbereiche: Diese bestehen aus einer Reihe paralleler Reihen von Löchern, üblicherweise durch eine zentrale Rinne getrennt. Die Löcher innerhalb einer Reihe sind elektrisch miteinander verbunden. Diese Anordnung eignet sich hervorragend für das Einsetzen von integrierten Schaltungen (ICs) und anderen Komponenten.

Die Stromleisten: Am Rand des Breadboards befinden sich in der Regel ein oder zwei Reihen von Löchern, die als Stromleisten dienen. Diese sind vertikal über die gesamte Länge des Breadboards miteinander verbunden und bieten eine bequeme Möglichkeit, Strom und Masse an verschiedenen Punkten der Schaltung zur Verfügung zu stellen.

Die Flexibilität eines Breadboards liegt in seiner Wiederverwendbarkeit und der Fähigkeit, Schaltungen ohne permanente Veränderungen aufzubauen. Dies macht es ideal für Experimente, da Fehler leicht korrigiert und Komponenten ohne Beschädigung entfernt werden können. Es ist auch ein ausgezeichnetes Lernwerkzeug, da es das Verständnis der Schaltungslogik und der Komponentenfunktionen in einer praktischen, visuellen Weise fördert.

Darüber hinaus sind Breadboards in verschiedenen Größen und mit unterschiedlichen Anzahlen von Verbindungspunkten erhältlich, um verschiedenen Anforderungen gerecht zu werden. Kleinere Breadboards eignen sich gut für einfache Projekte und Experimente, während größere für komplexere Schaltungen geeignet sind.

Trotz ihrer Vielseitigkeit haben Breadboards auch Einschränkungen. Sie eignen sich nicht gut für sehr hohe Frequenzen oder für Schaltungen, die eine hohe Leistung erfordern. Auch die Verbindungen können manchmal weniger zuverlässig sein als gelötete Verbindungen, besonders wenn das Breadboard mit der Zeit abgenutzt wird.

Insgesamt sind Breadboards ein unverzichtbares Werkzeug für jeden, der mit Elektronik arbeitet – von Anfängern, die die Grundlagen lernen, bis hin zu erfahrenen Entwicklern, die schnell und effizient Prototypen erstellen möchten. Sie sind das elektronische Äquivalent zu einem Skizzenbuch für einen Künstler: ein Ort, um Ideen zu erforschen und zu experimentieren, bevor das endgültige Werk entsteht.

5. PROJEKTE

Willkommen im Kapitel über innovative Elektronikprojekte mit dem Raspberry Pi Pico! In diesem Abschnitt werden Sie in eine breite Palette von Anwendungen eingeführt, die von der einfachen Ansteuerung von LEDs bis hin zur Entwicklung komplexerer Systeme wie automatisierten Wetterstationen und dynamischen Beleuchtungssystemen reichen. Jedes Projekt ist sorgfältig gestaltet, um Ihnen praktische Erfahrungen mit einer Vielzahl von Hardwarekomponenten zu bieten.

Beginnen Sie Ihre Entdeckungsreise mit grundlegenden Projekten, die Ihnen den Umgang mit GPIOs (General Purpose Input/Output) des Raspberry Pi Pico näherbringen, und steigern Sie Ihre Fähigkeiten mit fortgeschritteneren Themen wie der Steuerung von Servomotoren oder der Nutzung von Sensoren zur Umweltüberwachung. Mit Komponenten wie Drehencodern, Ultraschallsensoren, Buzzern und Neopixel-LEDs erlernen Sie das Design von interaktiven und reaktiven Systemen.

Jedes Projekt bietet eine detaillierte Einführung in die erforderlichen Komponenten, Schritt-für-Schritt-Anleitungen zur Hardwarekonfiguration und klare Beispiele für den Programmcode, die Ihnen helfen, die Prinzipien der Elektronik und der Computerprogrammierung zu verstehen. Darüber hinaus wird erläutert, wie externe Sensoren und Aktoren integriert werden können, um Echtzeitdaten zu erfassen und darauf zu reagieren.

Diese Projekte sind nicht nur lehrreich, sondern auch unterhaltsam und bieten zahlreiche Möglichkeiten zur Anpassung und Erweiterung, sodass Sie Ihre eigenen kreativen Lösungen entwickeln können. Egal, ob Sie ein Anfänger sind, der gerade erst anfängt, die Welt der digitalen Elektronik zu erkunden, oder ein erfahrener Entwickler, der seine Fähigkeiten erweitern möchte, dieses Kapitel bietet die nötigen Ressourcen und Inspirationen, um Ihre technischen Fähigkeiten zu verbessern und Spaß am Lernen zu haben. Bereiten Sie sich darauf vor, Ihre Programmier- und Elektronikfähigkeiten zu erweitern, während Sie durch jedes Projekt geführt werden und dabei Spaß am Erstellen und Experimentieren haben.

Die jeweiligen Beispielcodes finden Sie am Ende jedes Projektes. Sie können die Dateien aber auch **hier** herunterladen.

5. PROJEKTE

5.1 ENTFERNUNGSANZEIGE

In unserem ersten Projekt ist es unser Ziel, einen Ultraschall-Entfernungsmesser zu bauen, der Entfernungen auf unserem TFT-Display visualisiert. Dieses Projekt ist eine großartige Einführung in die Sensorik und Visualisieurng von Daten mit Ihrem Raspberry Pi Pico.

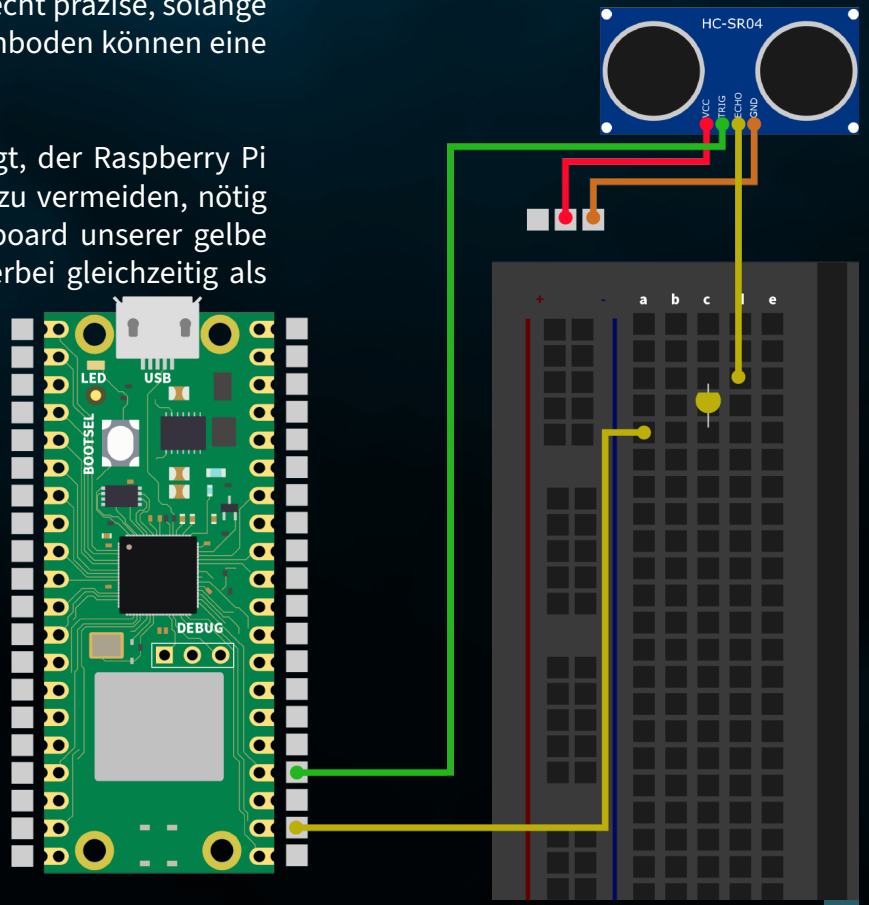
ULTRASCHALLSENSOR: Ein Sender sendet eine Ultraschallwelle aus und misst die Zeit, bis diese reflektiert wird und wieder bei ihm eintrifft. Da die Schallgeschwindigkeit in verschiedenen Medien wie Luft (343 m/s bei 20 °C) und Wasser (1.484 m/s) bekannt ist, lässt sich die Entfernung bis zur reflektierenden Fläche berechnen (halbieren der Laufzeit, da der Weg hin und zurück gemessen wurde). Ein Impuls vom Mikrocontroller am Trigger-Eingang löst eine Folge von acht kurzen Ultraschallimpulsen aus. Sobald das Signal wieder empfangen wurde, wird der Ausgang Echo kurzzeitig High. Die Zeit zwischen dem Trigger- und dem Echo-Signal entspricht der Laufzeit. Die Entfernungsmessung ist im Bereich von etwa 2 cm bis 400 cm möglich und recht präzise, solange die reflektierende Oberfläche möglichst hart und eben ist. Weiche Materialien wie Teppichboden können eine Messung sogar verhindern.

Da der Ultraschallsensor ein Sensor ist, der eine Spannungsversorgung von 5 V benötigt, der Raspberry Pi Pico jedoch nur 3,3 V Signale problemlos verarbeiten kann, ist es, um Beschädigungen zu vermeiden, nötig die Signalspannung herunterzuwandeln. Dies erreichen wir, indem wir auf dem Breadboard unserer gelbe LED in Reihe schalten und darüber unser Signal herunterwandeln. Die LED fungiert hierbei gleichzeitig als Signalanzeige für das aktive Echosignal.

Weitere Details zu LEDs findest du außerdem in Kapitel 5.5.

RASPBERRY PI PICO	ULTRASCHALLSENSOR
3V3	VCC
GP17	Trig
GP16	Echo
GND	GND

ACHTUNG! Für dieses Projekt ist es notwendig die Schalter für das **RELAY** und den **DHT11** auf **OFF** und den Schalter für das **TFT-DISPLAY** auf **ON** zu stellen.



ZUSAMMENFASSUNG: In unserem ersten Projekt messen wir Entfernungen mit dem Ultraschallsensor und visualisieren die gemessene Entfernung, indem wir die Grafik auf dem TFT-Display mehr oder weniger stark füllen. In unserem Beispiel füllen wir das Display ab einer gemessenen Entfernung von 100 cm vollständig.

```
# Bibliotheken laden
from machine import Pin, SPI
import ST7735
import time
import lcd_gfx

# Initialisierung von GPIO16 als Input und GPIO17 als Ausgang
trig = Pin(17, Pin.OUT)
echo = Pin(16, Pin.IN, Pin.PULL_DOWN)

# LCD initialisieren
spi = SPI(0, baudrate=8000000, polarity=0, phase=0, sck=Pin(18), mosi=Pin(19),
miso=Pin(16))
lcd = ST7735.ST7735(spi, rst=6, ce=17, dc=3)
backlight = Pin(2, Pin.OUT)
backlight.high()
lcd.reset()
lcd.begin()
lcd.fill_screen(lcd.rgb_to_565(255, 255, 255))

def translate(value, leftMin, leftMax, rightMin, rightMax):
    # Figure out how 'wide' each range is
    leftSpan = leftMax - leftMin
    rightSpan = rightMax - rightMin

    # Convert the left range into a 0-1 range (float)
    valueScaled = float(value - leftMin) / float(leftSpan)

    # Convert the 0-1 range into a value in the right range.
    return rightMin + (valueScaled * rightSpan)

# Endlosschleife zur Messung der Distanz
while True:
     # Abstandsmessung wird mittels des 10us langen Triggersignals gestartet
     trig.value(0)
     time.sleep(0.1)
     trig.value(1)

     # Nun wird am Echo-Eingang gewartet, bis das Signal aktiviert wurde
     # Danach wird die Zeit gemessen, wie lang es aktiviert bleibt
     time.sleep_us(2)
     trig.value(0)
     while echo.value()==0:
          pulse_start = time.ticks_us()
     while echo.value()==1:
          pulse_end = time.ticks_us()
     pulse_duration = pulse_end - pulse_start
```

Initialisierung des Ultraschall-sensors und des TFT-Displays

Hilfsfunktion zur Anpassung des Wertebereiches

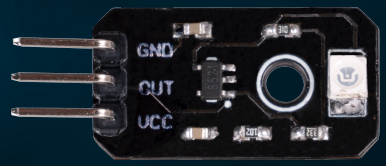
Distanzmessung

```
# Nun wird der Abstand mittels der aufgenommenen Zeit berechnet
distance = pulse_duration * 17165 / 1000000
distance = round(distance, 0)

# Serielle Ausgabe
print ('Distance:',"{:.0f}".format(distance),'cm')
time.sleep(1)

# Gemessenen Wert auf LCD Höhe angleichen
if(distance > 100):
    distance = 100
drawHeight = round(translate(distance, 0, 100, 0, 160))

# Fuelle das TFT Display
lcd.fill_screen(lcd.rgb_to_565(255, 255, 255))
lcd.draw_block(0, 0, 128, drawHeight, lcd.rgb_to_565(0, 255, 0))
```

Berechnung des Wertebereiches
und Beschreibung des TFT-Displays

5.2 WETTERSTATION

Tauchen Sie ein in das zweite Projekt unseres Elektronikabenteuers, bei dem Sie eine eigene Wetterstation erstellen! Dieses Projekt kombiniert den Einsatz eines UV-Sensors mit dem DHT11 Temperatur- und Feuchtigkeitssensor, um Ihnen nicht nur Einblicke in die aktuelle Wetterlage zu geben, sondern auch die UV-Strahlung an Ihrer Position zu messen. Auf dem farbenfrohen TFT-Display werden all diese Informationen anschaulich dargestellt, sodass Sie mit einem Blick das Wetter und die UV-Strahlung erfassen können.



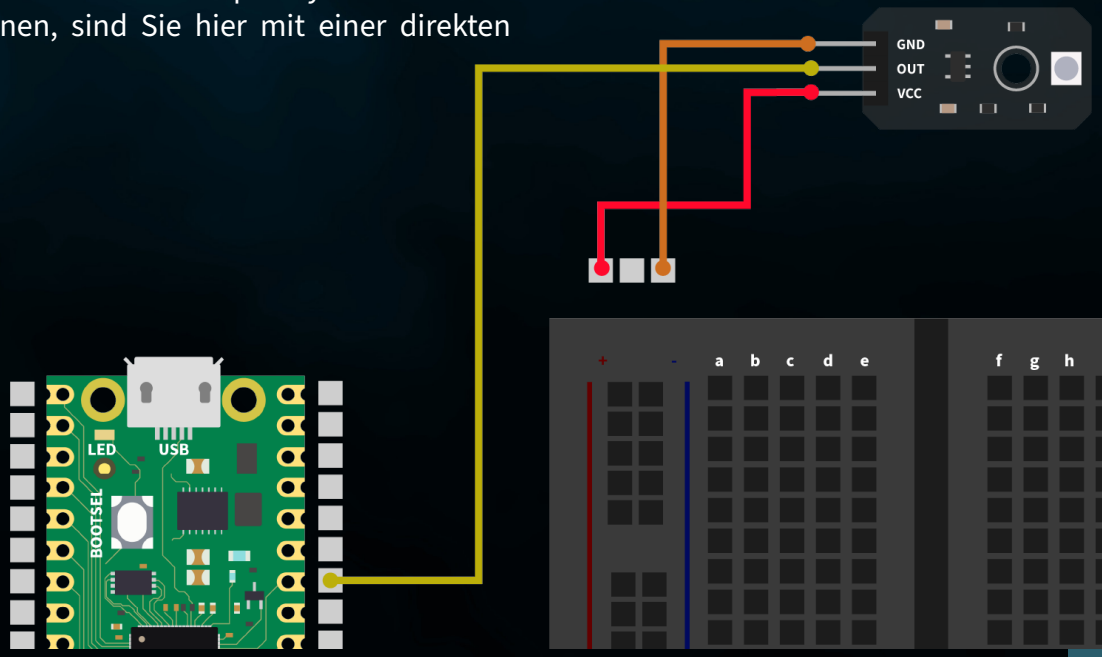
UV-SENSOR: Der UV-Sensor ist ein kleines Bauteil, welches uns dabei hilft, die unsichtbaren ultravioletten (UV) Strahlen der Sonne zu messen. UV-Strahlen sind jener Teil des Sonnenlichts, der zwar nicht sichtbar ist, aber Einfluss auf unsere Haut und Gesundheit haben kann. Denken Sie an Sonnenbrand oder die Bräunung der Haut - beides wird durch UV-Strahlen verursacht.

Im Kern enthält der Sensor ein Material, das auf UV-Strahlung reagiert. Wenn UV-Strahlen auf dieses Material treffen, verändert der Sensor seinen elektrischen Widerstand. Diese Veränderung wird vom Sensor in ein Signal umgewandelt, das wir messen und auslesen können. Mit Hilfe des Raspberry Pi Pico können wir dieses Signal dann in einen Wert umwandeln, der angibt, wie stark die UV-Strahlung gerade ist.

Verbinden Sie zunächst den UV-Sensor mit Hilfe der beiliegenden Kabel mit Ihrem Raspberry Pi Pico. Auch wenn Sie diesen natürlich auch auf das Breadboard aufstecken können, sind Sie hier mit einer direkten Kabelverbindung deutlich flexibler.

RASPBERRY PI PICO	UV-SENSOR
GND	GND
GP28	OUT
3V3	VCC

ACHTUNG! Für dieses Projekt ist es notwendig, den Schalter für das **RELAY** auf **OFF** und die Schalter für das **TFT-DISPLAY** und den **DHT11-SENSOR** auf **ON** zu stellen.



ZUSAMMENFASSUNG: Unsere Wetterstation ließt den DHT11- und den UV-Sensor aus und gibt die Daten auf dem TFT Display aus.

```
from machine import ADC, Pin, SPI
import utime
import dht
import ST7735  # Angenommen, dies ist die Bibliothek für Ihren TFT-Display

# DHT11 Sensor initialisieren
sensor_dht11 = dht.DHT11(Pin(0))

# UV-Sensor initialisieren
uv_sensor = ADC(2)  # Angenommen, GP28 ist ADC Pin Nummer 1 in Ihrer Konfiguration

# LCD initialisieren
spi = SPI(0, baudrate=8000000, polarity=0, phase=0, sck=Pin(18), mosi=Pin(19),
miso=Pin(16))
lcd = ST7735.ST7735(spi, rst=Pin(6), ce=Pin(17), dc=Pin(3))
backlight = Pin(2, Pin.OUT)
backlight.high()
lcd.reset()
lcd.begin()
lcd.fill_screen(lcd.rgb_to_565(255, 255, 255))

while True:
    lcd.fill_screen(lcd.rgb_to_565(255, 255, 255))

    # UV-Wert lesen
    uv_value = uv_sensor.read_u16()
    # Umrechnung in Prozent
    uv_percent = (uv_value / 65000) * 100
    print("UV Intensity (percent):", uv_percent)

    # DHT11 Werte lesen
    sensor_dht11.measure()
    temp = sensor_dht11.temperature()
    humid = sensor_dht11.humidity()

    # Werte auf LCD anzeigen
    lcd.p_string(20, 20, "Temp: {}C".format(temp))
    lcd.p_string(20, 40, "Humid: {}%".format(humid))
    lcd.p_string(20, 60, "UV: {:.2f}%".format(uv_percent))  # Anzeige der UV-
Intensität in Prozent mit zwei Nachkommastellen

    utime.sleep(10)
```

Initialisierung des TFT-Displays

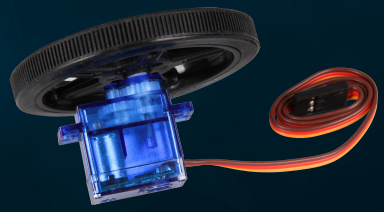
Messung der Sensorwerte

Ausgabe auf dem Display

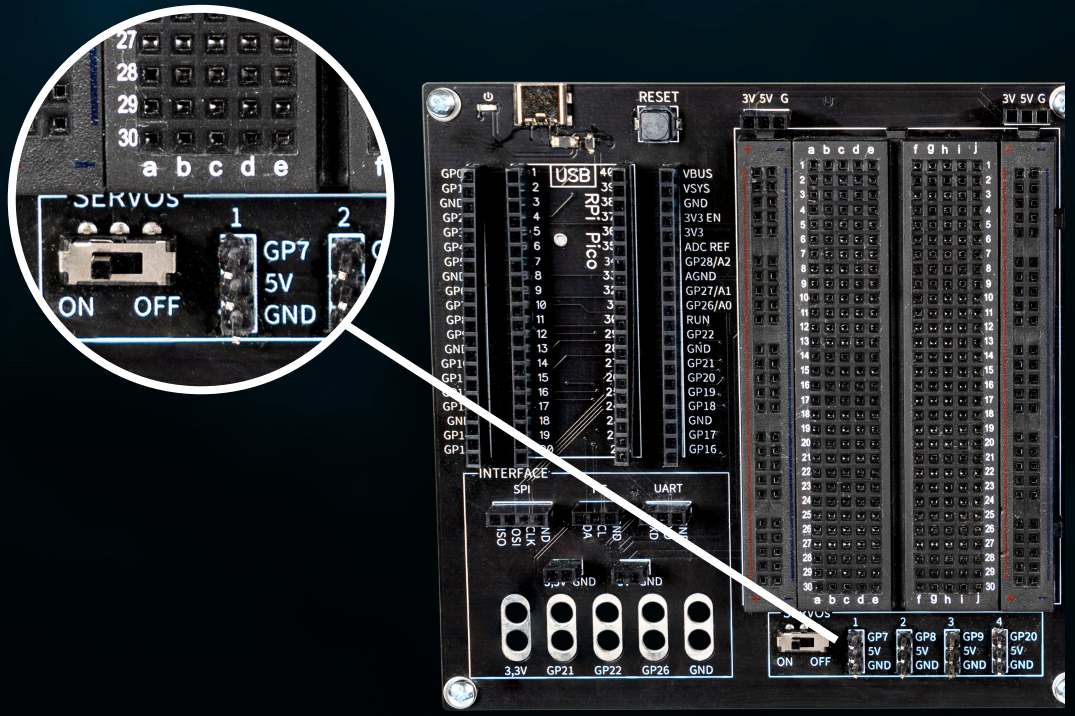
5.3 SERVOSTEUERUNG

Willkommen zum dritten Projekt in unserer Serie spannender Elektronikabenteuer mit dem Explorer Set! Diesmal dreht sich alles um Bewegung und Steuerung. Unser Ziel ist es, einen Servomotor so zu programmieren und anzusteuern, dass seine Drehrichtung durch einfache Knopfdrücke kontrolliert werden kann. Dieses Projekt bietet nicht nur eine hervorragende Einführung in die Welt der Motorensteuerung, sondern zeigt auch, wie man Interaktionen durch visuelle Rückmeldungen auf einem TFT-Display verstärken kann.

SERVOMOTOR: Ein Servo besteht aus einem Elektromotor mit Getriebe und Steuerelektronik. Auf der Ausgangsseite des Getriebes befindet sich ein Zahnrad, auf das das Servorad montiert wird. Einsatzgebiete für Servos sind zum einen der Modellbau, um bei einem Flugzeug oder Schiff beispielsweise die Flügel- oder Ruderstellung zu kontrollieren. Auch im Automobilbau werden immer mehr Servos verbaut, um Türen automatisch zu schließen, für Fensterheber, Spiegel und andere verstellbare Elemente.



Schließen Sie den Servomotor zunächst an die **SERVO-SCHNITTSTELLE** mit der Nummer **1** auf Ihrem Explorer Board an.



ACHTUNG! Für dieses Projekt ist es notwendig, den Schalter für das **TFT-DISPLAY**, die **BUTTONS** und die **SERVOS** auf **ON** zu stellen.

ZUSAMMENFASSUNG: Wir steuern unseren Servomotor und lassen ihn, gesteuert durch unsere Buttons, zwischen dem Links- und Rechtslauf wechseln.

Initialisierung des Servomotors und der Buttons

```
from machine import Pin, PWM
from utime import sleep

# Servo-Pin-Nummern
servoOnePin = 7

# Tasten-Pin-Nummern
buttonLeftPin = 11
buttonRightPin = 15

# Initialisierung des Servos
servoOne = PWM(Pin(servoOnePin))

# Initialisierung der Tasten mit PULL_UP, um den standardmäßigen HIGH-Zustand zu
nutzen
buttonLeft = Pin(buttonLeftPin, Pin.IN, Pin.PULL_UP)
buttonRight = Pin(buttonRightPin, Pin.IN, Pin.PULL_UP)

# Servo-Geschwindigkeiten in Nanosekunden
leftSpeed = 1700000  # Bewegt den Servo nach links
rightSpeed = 1300000  # Bewegt den Servo nach rechts

# Servofrequenz
servoOne.freq(50)  # Typische Servofrequenz von 50Hz

# Zustand des Servos
servoState = 'left'  # Startet mit Linkslauf

# Letzter Zustand der Buttons, um Flanken zu erkennen
lastButtonLeft = buttonLeft.value()
lastButtonRight = buttonRight.value()
```

Überprüfung der Buttons

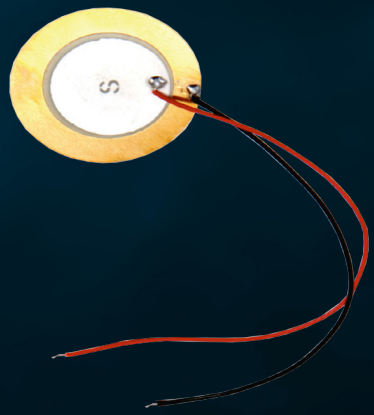
```
while True:
    # Aktuellen Zustand der Buttons auslesen
    currentButtonLeft = buttonLeft.value()
    currentButtonRight = buttonRight.value()

    # Überprüfen, ob eine Flanke von HIGH zu LOW erkannt wurde (Taste wurde
gedrückt)
    if lastButtonLeft == 1 and currentButtonLeft == 0:
        servoState = 'right'  # Ändert die Richtung nach rechts, wenn der linke
Button gedrückt wird
    elif lastButtonRight == 1 and currentButtonRight == 0:
        servoState = 'left'  # Ändert die Richtung nach links, wenn der rechte
Button gedrückt wird
```

```
# Aktualisieren der Zustände für den nächsten Durchlauf
lastButtonLeft = currentButtonLeft
lastButtonRight = currentButtonRight

# Steuert den Servo basierend auf dem aktuellen Zustand
if servoState == 'left':
    servoOne.duty_ns(leftSpeed)  # Bewegt den Servo nach links
else:
    servoOne.duty_ns(rightSpeed)  # Bewegt den Servo nach rechts

sleep(0.1)  # Kurze Pause für den Steuerzyklus
```

Servosteuerung

5.4 SELBSTGEBAUTER SUMMER

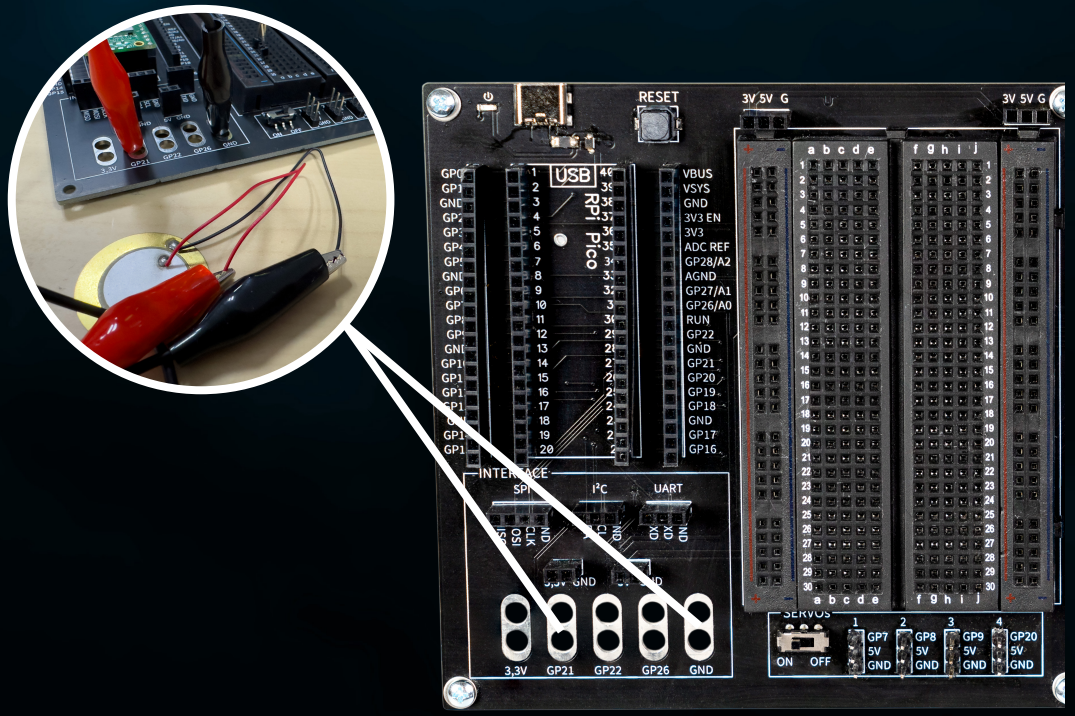
Im vierten Projekt unseres Elektronikabenteuers mit dem Explorer Set dreht sich alles um den Klang! Wir tauchen ein in die Welt der akustischen Signale, indem wir eine eigene Buzzer-Schaltung erstellen. Dieses Projekt bietet Ihnen nicht nur die Möglichkeit, die Grundlagen der Schaltkreiserstellung zu verstehen, sondern auch zu lernen, wie man mit einfachen Mitteln hörbare Signale erzeugt. Durch die Verwendung von Krokodilklemmen wird der Aufbau besonders benutzerfreundlich und zugänglich, auch für diejenigen, die noch am Anfang ihrer Elektronikreise stehen.



Zuerst verbinden wir den Buzzer sorgfältig mit dem Explorer Board, indem wir Krokodilklemmen nutzen. Diese Klemmen sind ideal für schnelle und flexible Verbindungen, ohne dass Lötarbeiten notwendig sind. Der Buzzer, ein kleines Bauteil, das in der Lage ist, Töne zu erzeugen, wird zum Mittelpunkt unseres Projekts. Durch das Anlegen von Strom vibriert der Buzzer und erzeugt so einen Ton.

Verbinden Sie zunächst eine Krokodilklemme mit dem GP21 Krokodilklemmen-Anschluss auf Ihrem Explorer Board. Das andere Ende der Krokodilklemme klemmen Sie an das rote Summerkabel. Eine weitere Krokodilklemme verbinden Sie mit dem GND Krokodilklemmen-Anschluss auf Ihrem Explorer Board. Das andere Ende der Klemme klemmen Sie an das schwarze Summerkabel.

RASPBERRY PI PICO	SUMMER
GP21	Rotes Kabel
GND	Schwarzes Kabel



ACHTUNG! Für dieses Projekt ist es notwendig, den Schalter für die **BUTTONS** auf **ON** zu stellen.

ZUSAMMENFASSUNG: Wir schließen einen externen Summer an unseren Raspberry Pi Pico an, um eine einfache, lustige Melodie zu spielen.

```
from machine import Pin, PWM
import utime

# Notenfrequenzen (in Hz)
notes = {
    'C4': 262,
    'D4': 294,
    'E4': 330,
    'F4': 349,
    'G4': 392,
    'A4': 440,
    'B4': 494,
    'C5': 523
}

# Melodie und Dauer (in ms)
melody = ['C4', 'D4', 'E4', 'C4', 'C4', 'D4', 'E4', 'C4', 'E4', 'F4', 'G4', 'E4',
'F4', 'G4']
durations = [500, 500, 500, 500, 500, 500, 500, 500, 500, 500, 1000, 500, 500,
1000]

# Initialisierung des Summers
buzzer = PWM(Pin(21))
buzzer.freq(440)  # Setze eine Startfrequenz

# Funktion, um eine Note zu spielen
def play_note(note, duration):
    if note in notes:
        buzzer.freq(notes[note])  # Setze die Frequenz basierend auf der Note
        buzzer.duty_u16(32767)  # Starte das PWM-Signal
        utime.sleep_ms(duration)  # Halte die Note für die Dauer
    buzzer.duty_u16(0)  # Stoppe das PWM-Signal (Note ausschalten)
    utime.sleep_ms(50)  # Kurze Pause zwischen den Noten

# Spiele die Melodie
for note, duration in zip(melody, durations):
    play_note(note, duration)

buzzer.deinit()  # Deaktiviere den PWM-Kanal, wenn fertig
```

Liste der Noten

Melodiespeicher

Funktion zum Abspielen der Noten

5.5 EIGENE SCHALTUNG

Im fünften Projekt unseres Elektronikabenteuers mit dem Explorer Set erforschen wir die faszinierende Welt der Lichtsteuerung. Diesmal bauen wir eine Schaltung auf, mit der Ihr LEDs steuern könnt. Dies bietet eine fantastische Möglichkeit, die Prinzipien der elektronischen Schaltungen zu verstehen und gleichzeitig die Kontrolle über Lichteffekte zu erlangen.

LEDS: LEDs, oder Leuchtdioden, sind kleine, aber leistungsstarke Lichtquellen, die in vielen elektronischen Projekten zum Einsatz kommen. Sie haben im Vergleich zu traditionellen Glühbirnen viele Vorteile, wie zum Beispiel eine längere Lebensdauer, einen geringeren Energieverbrauch und die Möglichkeit, in verschiedenen Farben zu leuchten. Eine LED besteht aus einem Halbleitermaterial, das Licht aussendet, wenn elektrischer Strom hindurchfließt.

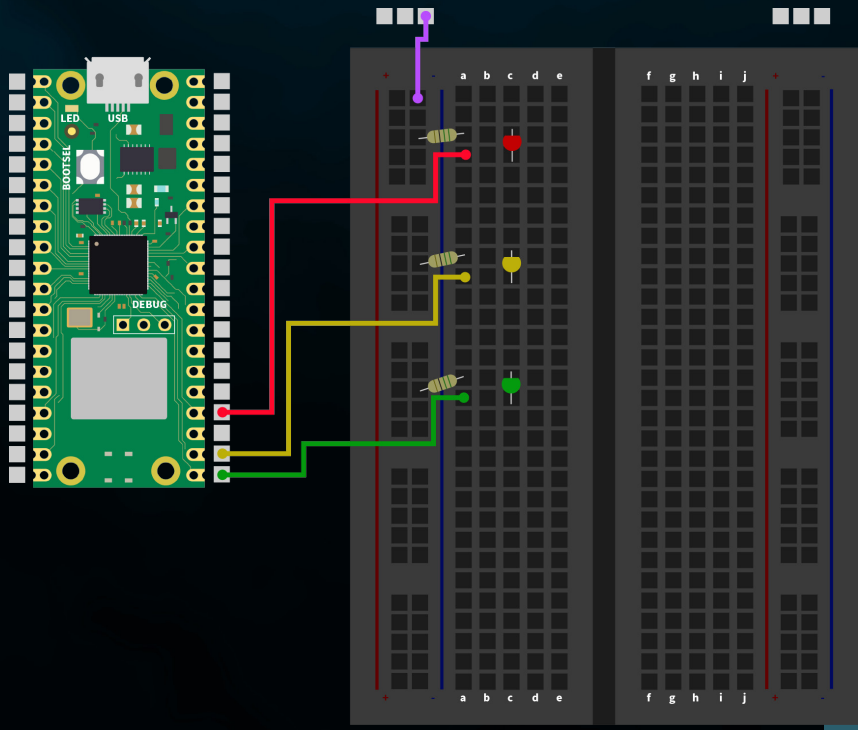
Die richtige Polung bei LEDs zu erkennen, ist wichtig, da sie nur funktionieren, wenn der Strom in der richtigen Richtung durch sie fließt. Das bedeutet, dass der positive Pol der Stromquelle mit dem positiven Ende der LED und der negative Pol der Stromquelle mit dem negativen Ende der LED verbunden werden muss.

So erkennen Sie die Polung einer LED:

Längeres Bein: Bei den meisten LEDs ist das längere Bein der Anode (+), also der positive Anschluss. Das kürzere Bein ist die Kathode (-), also der negative Anschluss.

Flache Kante: Auf der Seite des LED-Gehäuses kann eine flache Kante vorhanden sein. Diese Seite kennzeichnet in der Regel die Kathode, also den negativen Pol.

Bauen Sie die Schaltung zunächst wie in der folgenden Grafik nach. Aber achten Sie unbedingt auf die richtige Polung der LEDs. Verwenden Sie außerdem für jede LED einen 56 Ω (Grün-Blau-Schwarz) Vorwiderstand.



RASPBERRY PI PICO	LEDS
GP18	Rote LED
GP17	Gelbe LED
GP16	Grüne LED

ACHTUNG! Für dieses Projekt ist es notwendig, den Schalter für das **TFT-DISPLAY** auf **OFF** zu stellen.

ZUSAMMENFASSUNG: Wir steuern drei verschiedene LEDs über die Pins des Raspberry Pi Pico an, wobei jede LED abwechselnd blinkt.

```
from machine import Pin
import utime

# Initialisiere die LEDs
red_led = Pin(18, Pin.OUT)
yellow_led = Pin(17, Pin.OUT)
green_led = Pin(16, Pin.OUT)

# Blinkfunktion für eine LED
def blink_led(led, duration):
    led.value(1)  # LED einschalten
    utime.sleep(duration)
    led.value(0)  # LED ausschalten
    utime.sleep(duration)

# Hauptschleife
while True:
    blink_led(red_led, 0.5)     # Rote LED blinkt für 0.5 Sekunden
    blink_led(yellow_led, 0.5) # Gelbe LED blinkt für 0.5 Sekunden
    blink_led(green_led, 0.5)   # Grüne LED blinkt für 0.5 Sekunden
```

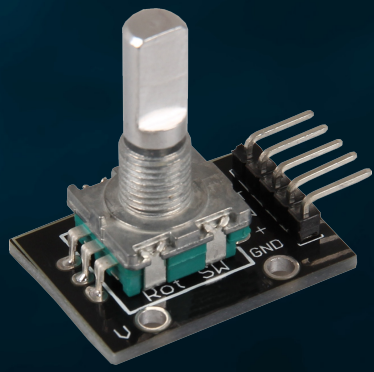
Initialisierung der LEDs

LED Blinkfunktion

Hauptschleife

5.6 LED-STEUERUNG

Im sechsten Projekt unseres Elektronik-Abenteuers nutzen wir unseren Drehencoder, um die Helligkeit und Farbe von LEDs zu steuern – eine einfache, doch faszinierende Art, in die Elektronik einzutauchen. Der Drehencoder, unser zentrales Bedienelement, ermöglicht durch seine doppelte Funktion eine spielerische Interaktion. Helligkeitsänderungen werden über die Drehung gesteuert, während das Drücken des Encoders die LED-Farben durchläuft – ideal, um die Grundlagen der Elektronik und Farbmischung zu veranschaulichen.

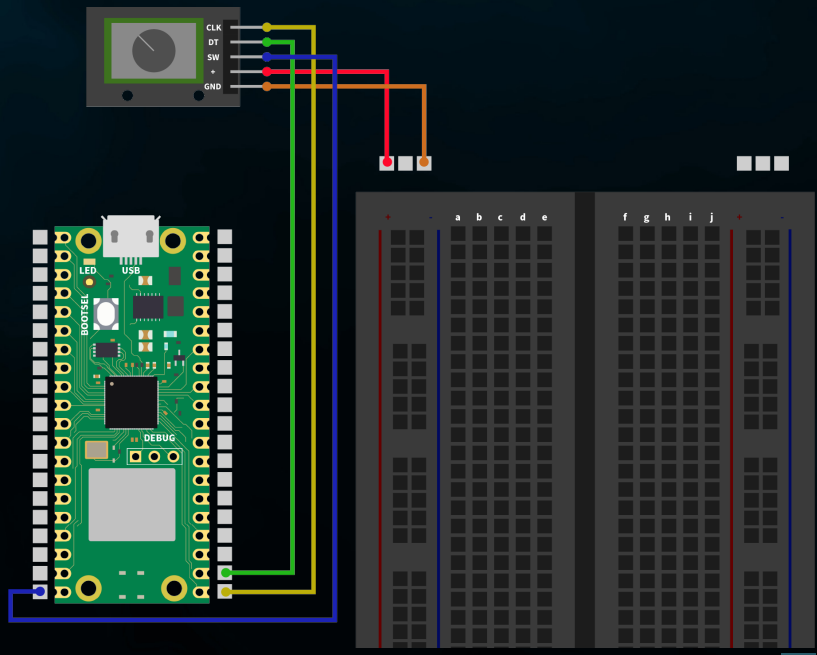


DREHENCODER: Der Drehencoder ist ein cleveres kleines Gerät, das Ihre Drehbewegungen in elektronische Signale umwandelt. Stellen Sie sich einen Drehknopf vor, wie Sie ihn von einem Radio kennen. Wenn Sie diesen Knopf drehen, kann der Drehencoder messen, wie weit und in welche Richtung Sie gedreht haben. Diese Information kann dann verwendet werden, um zum Beispiel die Lautstärke zu ändern, durch Menüs zu navigieren oder, in unseren Projekten, die Helligkeit von LEDs anzupassen.

Drehencoder haben oft auch einen eingebauten Knopf – das bedeutet, sie können auch als Druckschalter fungieren. Wenn Sie den Drehknopf drücken, erkennt der Drehencoder diesen Druck als separates Signal. Dies kann für verschiedene Funktionen genutzt werden, wie zum Beispiel das Ein- und Ausschalten eines Geräts oder das Wechseln von Betriebsmodi.

ZUSAMMENGEFASST: Ein Drehencoder erlaubt es Ihnen, durch Drehen und Drücken verschiedene Befehle an Ihre Elektronikprojekte zu senden. Er ist ein intuitives und vielseitiges Werkzeug, das Interaktionen mit Ihren Projekten einfach und spaßig macht.

Schließen Sie zunächst den Drehencoder wie folgt an Ihren Raspberry Pi Pico an:



RASPBERRY PI PICO	DREHENCODER
GP16	CLK
GP17	DT
GP15	SW
3V3	+
GND	GND

ACHTUNG! Für dieses Projekt ist es notwendig, den Schalter für das **TFT-DISPLAY** auf **OFF** und den Schalter für die **LEDS** auf **ON** zu stellen.

ZUSAMMENFASSUNG: Wir verwenden den Drehencoder, um die Farbe und Helligkeit unserer vier LEDs zu steuern. Durch das Drehen des Encoders wechseln die Helligkeit, während ein Druck auf den Encoder die Farbe der LEDs anpasst.

```
from machine import Pin
import utime
import neopixel

# Neopixel setup
NUM_LEDS = 4
PIXEL_PIN = 1
np = neopixel.NeoPixel(Pin(PIXEL_PIN), NUM_LEDS)

# Drehencoder setup
PIN_CLK = Pin(16, Pin.IN, Pin.PULL_UP)
PIN_DT = Pin(17, Pin.IN, Pin.PULL_UP)
BUTTON_PIN = Pin(15, Pin.IN, Pin.PULL_UP)

# Globale Variablen
counter = 0
PIN_CLK_LAST = PIN_CLK.value()
delayTime = 0.001
debounce_time_encoder = 0
debounce_time_button = 0

# Farben initialisieren
colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 255)]  # Rot, Grün,
Blau, Weiß
color_index = 0

# Helligkeit initialisieren
brightness_levels = [0.2, 0.4, 0.6, 0.8, 1.0]
brightness_index = 0

def update_leds(color, brightness):
    dimmed_color = tuple([int(c * brightness) for c in color])
    for i in range(NUM_LEDS):
        np[i] = dimmed_color
    np.write()

def rotaryFunction(null):
    global counter, brightness_index, debounce_time_encoder
    PIN_CLK_CURRENT = PIN_CLK.value()
    if PIN_CLK_CURRENT != PIN_CLK_LAST and (utime.ticks_ms() - debounce_time_
encoder) > 300:
        if PIN_DT.value() != PIN_CLK_CURRENT:
            brightness_index = (brightness_index + 1) % len(brightness_levels)
        else:
            brightness_index = (brightness_index - 1) % len(brightness_levels)
        update_leds(colors[color_index], brightness_levels[brightness_index])
        debounce_time_encoder = utime.ticks_ms()
```

Initialisierung der LEDs und des Drehencoders

Funktion für den Drehencoder

```
def counterReset(null):
    global color_index, debounce_time_button
    if (utime.ticks_ms() - debounce_time_button) > 300:
        color_index = (color_index + 1) % len(colors)
        update_leds(colors[color_index], brightness_levels[brightness_index])
        debounce_time_button = utime.ticks_ms()



PIN_CLK.irq(trigger=Pin.IRQ_FALLING | Pin.IRQ_RISING, handler=rotaryFunction)
BUTTON_PIN.irq(trigger=Pin.IRQ_FALLING, handler=counterReset)

update_leds(colors[color_index], brightness_levels[brightness_index])

while True:
    utime.sleep(delayTime)
```

Liste der Noten

5.7 AUTOMATISCHE HELLIGKEITSSTEUERUNG

Im siebten Projekt unseres Elektronik-Abenteuers verwenden wir eine Fotodiode, um die Helligkeit von LEDs automatisch zu steuern. Die Fotodiode wandelt Licht in ein elektrisches Signal um, sodass die LEDs heller leuchten, wenn es dunkel ist, und gedimmt werden, wenn mehr Umgebungslicht vorhanden ist.

Durch die Verbindung der Fotodiode mit dem Explorer Board und entsprechende Programmierung auf dem Raspberry Pi Pico passen sich die LEDs intelligent an die Umgebungshelligkeit an. Dieses Projekt zeigt, wie man mit einfachen Komponenten reaktive und energiesparende Elektroniksysteme bauen kann.

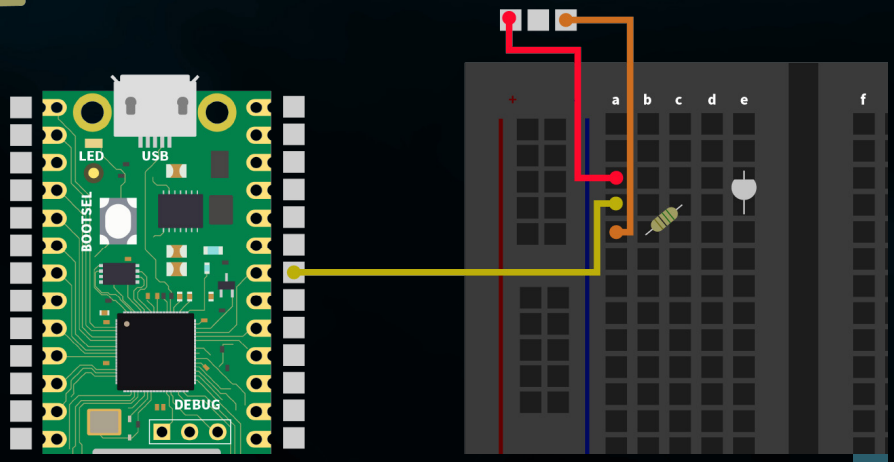
FOTODIODE: Eine Fotodiode ist ein spezieller Typ von Halbleiter, der darauf reagiert, wenn Licht auf ihn trifft, indem er elektrischen Strom erzeugt. Denken Sie an eine Fotodiode wie an ein kleines Solarpanel: Wenn Licht darauf fällt, wandelt sie dieses Licht in ein elektrisches Signal um. Je mehr Licht die Fotodiode erreicht, desto stärker wird das Signal.

Fotodioden sind sehr empfindlich und können sogar geringe Lichtmengen erkennen, was sie ideal für Projekte macht, bei denen es darauf ankommt, die Helligkeit oder das Vorhandensein von Licht zu messen. Sie können zum Beispiel in automatischen Helligkeitsreglern, Lichtsensoren oder als Teil eines Systems zur Steuerung von Lichtverhältnissen eingesetzt werden.

Kurz gesagt: Fotodioden sind effektive Lichtdetektoren, die es uns ermöglichen, elektronische Geräte intelligent auf Änderungen in der Umgebungsbeleuchtung reagieren zu lassen.

Schließen Sie zunächst die Fotodiode wie folgt über das Breadboard an. Bitte beachten Sie, dass auch hier die Verwendung eines Widerstandes erforderlich ist. Verwenden Sie hier den 100 kΩ (Braun-Schwarz-Gelb) Widerstand.

RASPBERRY PI PICO	FOTODIODE
3V3	Positive Kathode
GP28/A2	Negative Anode



ACHTUNG! Für dieses Projekt ist es notwendig, den Schalter für das **RELAIS** auf **OFF** und den Schalter für die **LEDS** auf **ON** zu stellen.

ZUSAMMENFASSUNG: Wir verwenden unsere Fotodiode, um die Umgebungshelligkeit zu messen und die Helligkeit von vier LEDs anzupassen. Die Intensität der LEDs ändert sich entsprechend dem von der Fotodiode erfassten Licht, wobei dunklere Umgebungen zu helleren LEDs führen und umgekehrt. Verwenden Sie am besten eine Taschenlampe um das bestmögliche Ergebnis zu erzielen.

```
from machine import Pin, ADC
import neopixel
import utime

# Neopixel setup
NUM_LEDS = 4
PIXEL_PIN = 1
np = neopixel.NeoPixel(Pin(PIXEL_PIN), NUM_LEDS)

# Fotodiode setup auf ADC Pin GP28 (A2)
fotodiode = ADC(2)

# Umwandlungsfunktion für Helligkeitswerte der Fotodiode in eine geeignete
Helligkeit für die LEDs
def brightness_from_light(sensor_value):
    # Normierung des Sensorwertes (0 bis 65535) auf Helligkeitsstufen (0.1 bis
1.0)
    return max(0.1, min(1.0, sensor_value / 65535))

# Standardfarbe
color = (255, 255, 255)  # Weiß

# LED-Update-Funktion
def update_leds(brightness):
    dimmed_color = tuple([int(c * brightness) for c in color])
    for i in range(NUM_LEDS):
        np[i] = dimmed_color
    np.write()

while True:
    # Lies den Sensorwert von der Fotodiode
    light_value = fotodiode.read_u16()
    print(light_value)

    # Berechne die Helligkeit basierend auf dem Sensorwert
    brightness = brightness_from_light(light_value)

    # Aktualisiere die LEDs mit der neuen Helligkeit
    update_leds(brightness)

    # Wartezeit zur Entlastung der CPU und für flüssigere Helligkeitsübergänge
    utime.sleep(0.5)
```

Initialisierung der LEDs und der Fotodiode

Messung der Fotodiode und Steuerung der Helligkeit

5.8 RGB-LED STEUERUNG

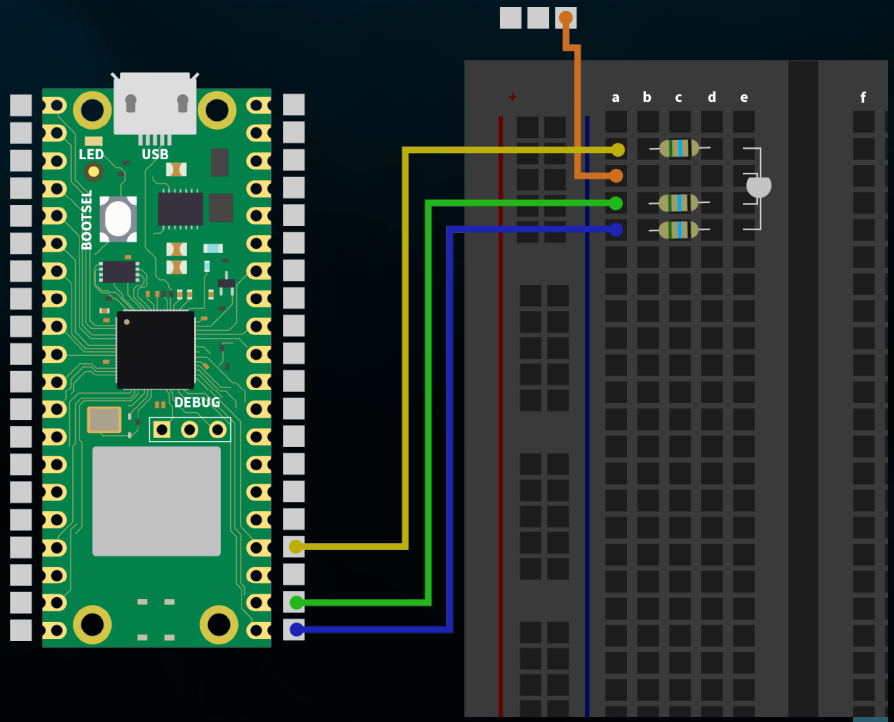
Im achten und letzten Projekt unserer Elektronik-Reihe widmen wir uns der Farbsteuerung der RGB-LEDs mittels der integrierten Buttons am Explorer Board. RGB-LEDs sind besondere Leuchtdioden, die Rot, Grün und Blau (RGB) Licht kombinieren, um eine breite Palette von Farben darzustellen. Durch das individuelle Einstellen der Intensität jeder Farbkomponente können wir fast jede Farbe erzeugen.

In diesem Projekt schließen wir die RGB-LED an das Breadboard an und nutzen die vorhandenen Buttons, um die Farben der LED zu steuern. Jeder Button ist einer Farbe (Rot, Grün, Blau) zugeordnet.

RGB LED: Eine RGB-LED kombiniert Rot, Grün und Blau in einem einzigen Lichtpunkt. Durch Verändern der Helligkeit jeder der drei Farben können fast alle Farben erzeugt werden. Dies geschieht durch Pulsweitenmodulation (PWM), welche die Intensität jeder Farbe steuert. So ermöglichen RGB-LEDs mit nur drei Farben ein breites Farbspektrum, ideal für bunte Beleuchtungsprojekte.

Schließen Sie zunächst die RGB-LED wie folgt an das Breadboard an. Beachten Sie unbedingt, dass auch hier jeder der drei Farbkanäle einen Vorwiderstand benötigt. Sie sollten hier den 56 Ω Widerstand (Grün-Blau-Schwarz) verwenden.

RASPBERRY PI PICO	RGB-LED
GP18	Erster Pin
GND	Zweiter Pin
GP17	Dritter Pin
GP16	Vierter Pin



ACHTUNG! Für dieses Projekt ist es notwendig, den Schalter für das **TFT** auf **OFF** und für die **BUTTONS** auf **ON** zu stellen.

ZUSAMMENFASSUNG: Die drei Farbkanäle der RGB-LED (Rot, Grün & Blau) werden durch die Buttons (Links, Oben & Rechts) an bzw. ausgeschaltet.

```python
from machine import Pin
import utime

# Initialisiere die LED-Pins
red_led = Pin(18, Pin.OUT)
green_led = Pin(17, Pin.OUT)
blue_led = Pin(16, Pin.OUT)

# Initialisiere die Button-Pins
button_red = Pin(15, Pin.IN, Pin.PULL_UP)
button_green = Pin(10, Pin.IN, Pin.PULL_UP)
button_blue = Pin(11, Pin.IN, Pin.PULL_UP)

# Zustände der LEDs speichern
red_state = False
green_state = False
blue_state = False

def toggle_led(led, state):
    led.value(state)

while True:
    # Überprüfe den Zustand des roten Buttons
    if button_red.value() == 0:
        red_state = not red_state
        toggle_led(red_led, red_state)
        utime.sleep(0.2)  # Entprellung

    # Überprüfe den Zustand des grünen Buttons
    if button_green.value() == 0:
        green_state = not green_state
        toggle_led(green_led, green_state)
        utime.sleep(0.2)  # Entprellung

    # Überprüfe den Zustand des blauen Buttons
    if button_blue.value() == 0:
        blue_state = not blue_state
        toggle_led(blue_led, blue_state)
        utime.sleep(0.2)  # Entprellung
```

Initialisierung der LED und der Buttons

Prüfung der Buttons und Steuerung der LED

6. INFORMATIONS- & RÜCKNAHMEPFLICHTEN

UNSERE INFORMATIONS- UND RÜCKNAHMEPFLICHTEN NACH DEM ELEKTROGESETZ (ELEKTROG)

SYMBOL AUF ELEKTRO- UND ELEKTRONIKGERÄTEN:
Diese durchgestrichene Mülltonne bedeutet, dass Elektro- und Elektronikgeräte nicht in den Hausmüll gehören. Sie müssen die Altgeräte an einer Erfassungsstelle abgeben. Vor der Abgabe haben Sie Altbatterien und Altakkumulatoren, die nicht vom Altgerät umschlossen sind, von diesem zu trennen.

RÜCKGABEMÖGLICHKEITEN:
Als Endnutzer können Sie beim Kauf eines neuen Gerätes, Ihr Altgerät (das im Wesentlichen die gleiche Funktion wie das bei uns erworbene neue erfüllt) kostenlos zur Entsorgung abgeben. Kleingeräte bei denen keine äußere Abmessung größer als 25 cm sind, können unabhängig vom Kauf eines Neugerätes in haushaltsüblichen Mengen abgeben werden.

MÖGLICHKEIT RÜCKGABE AN UNSEREM FIRMENSTANDORT WÄHREND DER ÖFFNUNGSZEITEN:
SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

MÖGLICHKEIT RÜCKGABE IN IHRER NÄHE:
Wir senden Ihnen eine Paketmarke zu, mit der Sie das Gerät kostenlos an uns zurücksenden können. Hierzu wenden Sie sich bitte per E-Mail an service@joy-it.net oder per Telefon an uns.

INFORMATIONEN ZUR VERPACKUNG:
Verpacken Sie Ihr Altgerät bitte transportsicher, sollten Sie kein geeignetes Verpackungsmaterial haben oder kein eigenes nutzen möchten kontaktieren Sie uns, wir lassen Ihnen dann eine geeignete Verpackung zukommen.

7. SUPPORT

Wir sind auch nach dem Kauf für Sie da. Sollten noch Fragen offen bleiben oder Probleme auftauchen, stehen wir Ihnen auch per E-Mail, Telefon und Ticket-Supportsystem zur Seite.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Telefon: +49 (0)2845 9360 – 50 (Mo. - Do.: 09:00 - 17:00 Uhr, Fr: 09:00 - 14:30 Uhr)

Für weitere Informationen besuchen Sie unsere Website:

WWW.JOY-IT.NET