EXPLORER SET

RB-P-XPLR-SET

ÍNDICE

1. INFORMAÇÕES GERAIS

Caro cliente, obrigado por ter escolhido o nosso produto. A seguir, mostramos-lhe o que deve ter em conta durante a colocação em funcionamento e a utilização.

Se tiver algum problema inesperado durante a utilização, não hesite em contactar-nos.

2. VISÃO GERAL DO DISPOSITIVO E ATRIBUIÇÃO DE PINOS

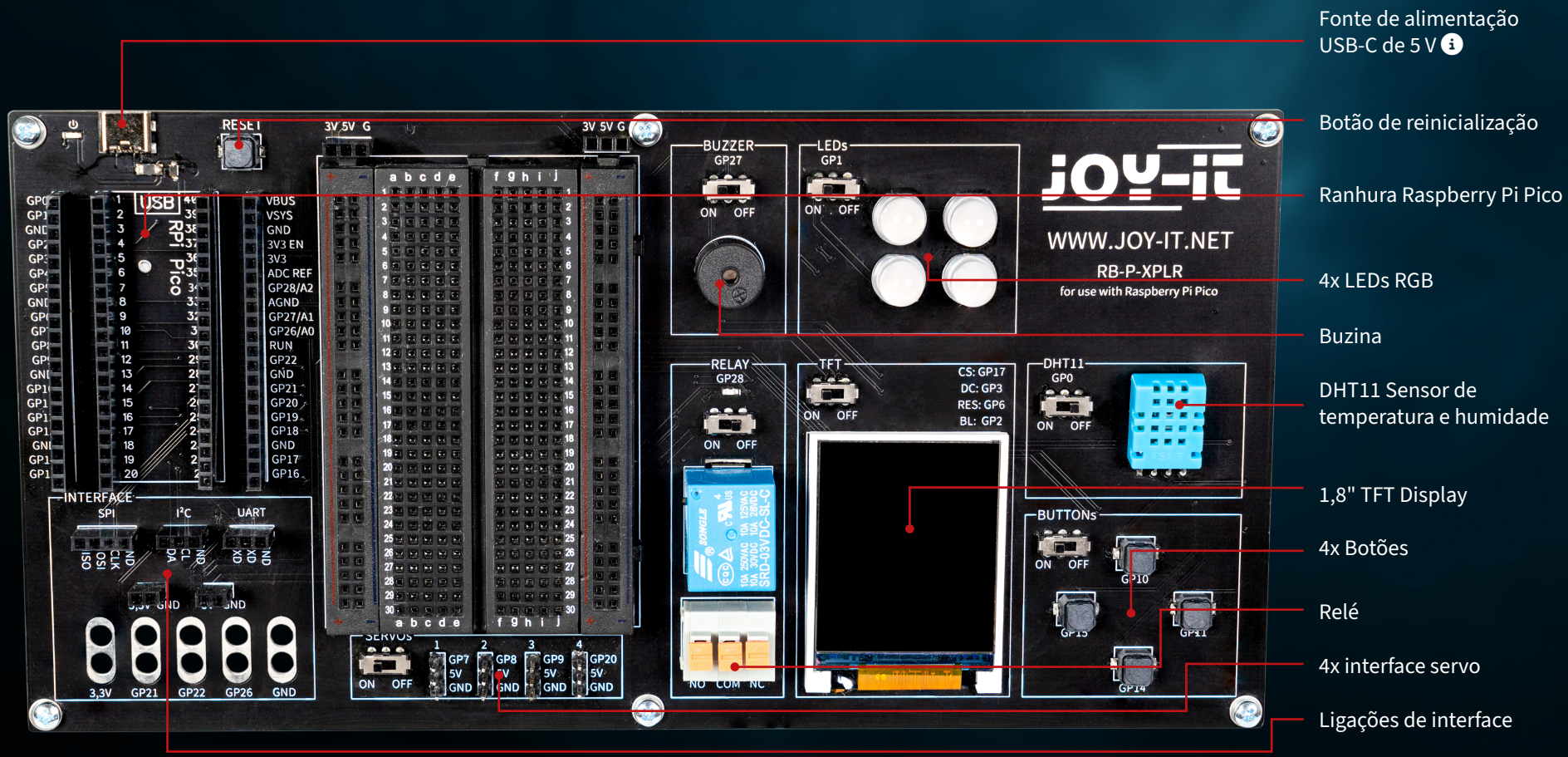
A nossa placa Explorer é a forma mais simples e eficiente de desenvolver os seus projectos Raspberry Pi Pico.

Com os componentes mais importantes já integrados, poupa tempo e esforço na instalação dos cabos. A placa Explorer tem uma vasta gama de conectores de interface para que possa ligar os seus projectos a uma variedade de módulos e dispositivos. Com a placa de ensaio integrada, pode construir e realizar rapidamente os seus próprios projectos.

Graças à possibilidade de ligar ou desligar todos os módulos individualmente, pode utilizar os seus pinos, que também são encaminhados separadamente para o exterior, para outros projectos ou fazer experiências na placa de ensaio integrada em qualquer altura.

Todos os componentes integrados podem ser desligados através do respetivo interrutor, se não forem necessários. Isto significa que os pinos associados também podem ser utilizados para outros componentes, se necessário.

À esquerda e à direita do Raspberry Pi Pico, todos os pinos são projectados adicionalmente. Os componentes podem ser ligados diretamente aqui ou encaminhados para a placa de ensaio integrada através de cabos adicionais.

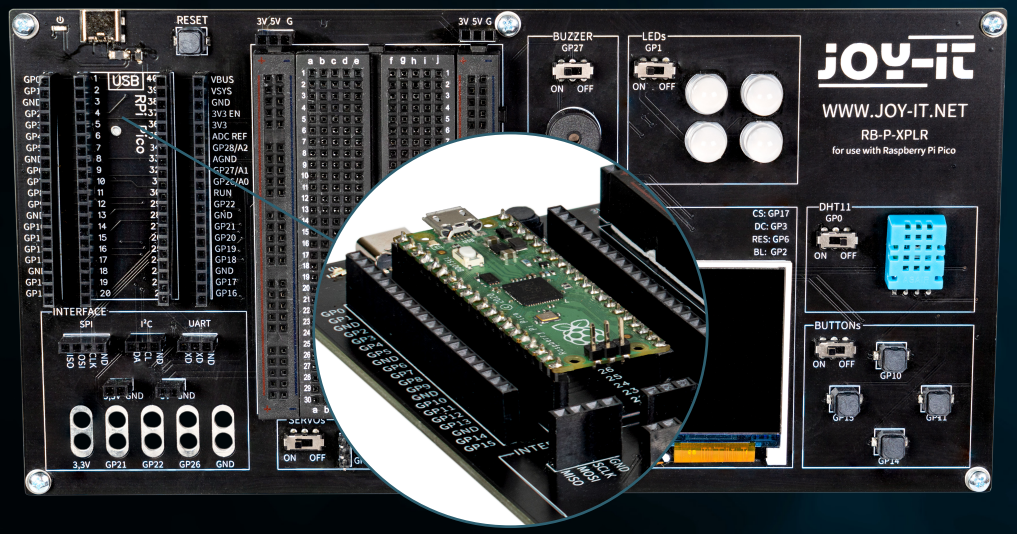


Tenha em atenção que a ligação USB-C deve estar sempre ligada para ser utilizada. Não é possível uma fonte de alimentação através da ligação micro USB do Raspberry Pi Pico.

ATRIBUIÇÃO DE PINOS	
Buzzer	GP27
LEDs	GP1
Relay	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

Em primeiro lugar, ligue o Raspberry Pi Pico à ranhura da sua placa.



Agora, ligue um cabo micro USB ao seu computador e ao Raspberry Pi Pico para programação.

ATENÇÃO! A porta USB-C na placa Explorer é usada exclusivamente para fornecimento de energia. Não é usada para transferir dados para o Raspberry Pi. Pode utilizar um programa de desenvolvimento adequado à sua escolha para transferir o nosso programa de exemplo. Recomendamos o **Thonny Python IDE**.

ATENÇÃO! Se é novo no mundo dos microcontroladores e da eletrónica, não se preocupe! Preparámos um guia de iniciação especial para si. Este guia foi especialmente concebido para as necessidades dos principiantes e explica como utilizar o Raspberry Pi Pico passo a passo.

Desde a configuração básica até à execução de projectos, neste guia iremos acompanhá-lo ao longo de todo o processo. O nosso guia inclui explicações fáceis de compreender e dicas úteis para o ajudar a desenvolver rápida e eficazmente as suas competências à escala com o Raspberry Pi Pico. Pode descarregar o nosso **guia aqui**.

4. MÓDULOS EM PORMENOR

De seguida, todos os módulos disponíveis na placa Explorer são explicados individualmente com códigos de exemplo. Aqui pode descarregar todos os códigos de exemplo e bibliotecas, bem como um código de exemplo que liga todos os módulos entre si.

Para a utilização de alguns módulos, são utilizadas bibliotecas externas e um ficheiro de fontes. Descarregue as bibliotecas e carregue-as na pasta lib do seu Raspberry Pi Pico. Coloque o ficheiro de fontes no diretório raiz do seu Raspberry Pi Pico.

4.1 BUZINA

Uma campainha produz um sinal sonoro, semelhante a um altifalante. No entanto, ao contrário de um altifalante, só é adequado para uma gama de frequências limitada, pelo que não produz um bom som para reproduzir música ou fala. No entanto, é ideal para gerar sinais sonoros de aviso sob a forma de bips. Sempre que um dispositivo elétrico gera um sinal de aviso, é quase sempre uma campainha. Por exemplo, em despertadores, detectores de fumo ou no lembrete do cinto de segurança nos automóveis.

A campainha está ligada ao pino GPIO GP27.

```
# 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 LEDS RGB

Os LEDs RGB são um tipo de díodo emissor de luz que combina vermelho, verde e azul para produzir uma variedade de cores. Tal como uma campainha apenas produz tons simples, os LED RGB não podem apresentar imagens complexas, mas são excelentes na mistura e variação de cores. Cada LED numa unidade RGB pode ser variado em intensidade para produzir diferentes tonalidades, desde pastéis suaves a cores brilhantes e saturadas. Isto torna-os ideais para iluminação ambiente, iluminação decorativa e em aplicações onde são necessários sinais visuais, como em configurações de jogos ou como indicadores de estado em dispositivos electrónicos. A sua versatilidade e eficiência energética tornaram-nos uma escolha popular nos sistemas de iluminação modernos, embora, tal como a campainha, o seu funcionamento simples signifique que não podem criar imagens ou padrões complexos sem unidades de controlo adicionais.

Os LEDs GPIO estão ligados ao pino GPIO GP1.

```
# 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 RELÉ

Os relés são alguns dos componentes electromecânicos mais antigos e funcionam como interruptores controlados eletricamente. Com uma pequena tensão de entrada e baixa corrente, uma grande carga eléctrica pode ser ligada e desligada na saída. Quando o relé comuta, o LED vermelho também se acende. Pode inserir extremidades de cabos descarnados na tomada de terminais (premindo a alavanca laranja) para utilizar as três ligações.

O relé está ligado ao pino GPIO GP28.

```
# 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

O ecrã de cristais líquidos (LCD TFT) com cerca de 65.000 cores e uma diagonal de 1,8 polegadas tem uma resolução de 128×160 pixels e pode ser controlado através de SPI. É adequado para a apresentação de gráficos e imagens coloridos. As letras e outros caracteres são apresentados como gráficos constituídos por muitos pontos individuais.

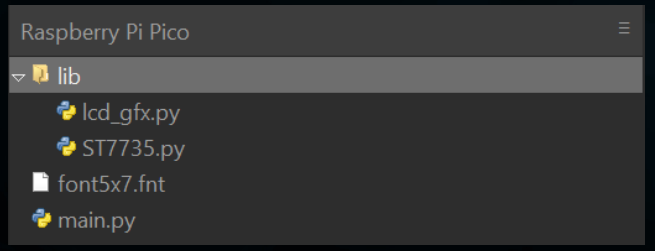
O TFT está ligado aos pinos GPIO GP17 (CS), GP3 (DC), GP6 (RES) e GP2 (BL).

```
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!‘)
```

Para além dos textos, também podem ser apresentados rectângulos, por exemplo:

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

ATENÇÃO! São necessários dois ficheiros de biblioteca separados e um ficheiro de fonte para o ecrã TFT; pode descarregar os ficheiros necessários aqui. Em seguida, transfira todos os ficheiros da pasta Libraries para o diretório raiz do seu Raspberry Pi Pico, de modo a que a estrutura de pastas tenha o seguinte aspeto



4.5 DHT 11

O sensor DHT11 pode detetar temperaturas de 0 °C a 50 °C (precisão de ±2 °C) e humidade relativa de 20 % a 80 % (±5 %) (no máximo uma vez por segundo). As estações meteorológicas são provavelmente a principal área de aplicação de um sensor como o DHT11. Para testar a funcionalidade, basta aproximar a boca do sensor e expirar lentamente. O ar respirado difere do ambiente em termos de temperatura e humidade, o que deve levar a uma alteração significativa dos valores.

O DHT11 está ligado ao pino GPIO GP0.

```
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 BOTÕES

Os botões são elementos interactivos nas interfaces de utilizador que cumprem uma função simples mas essencial: a introdução de dados pelo utilizador. Tal como os LED RGB podem apresentar uma variedade de cores, os botões são utilizados para iniciar uma vasta gama de comandos e acções em ambientes digitais.

Os botões estão ligados aos pinos GPIO GP10 (em cima), GP11 (à direita), GP14 (em baixo) e GP15 (à esquerda).

```
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

Um servo é constituído por um motor elétrico com caixa de velocidades e eletrónica de controlo. No lado de saída da caixa de velocidades existe uma roda dentada na qual está montada a buzina do servo. Os servos são utilizados na construção de modelos, por exemplo, para controlar a posição da asa ou do leme de um avião ou navio. Cada vez mais servos são também utilizados na engenharia automóvel para fechar automaticamente portas, para reguladores de janelas, espelhos e outros elementos ajustáveis.

As ligações do servo são os pinos GPIO GP7, GP8, GP9 e 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

As ligações de interface desempenham um papel crucial no mundo da eletrónica, à semelhança dos botões nas interfaces de utilizador. Permitem a comunicação e a alimentação de energia entre diferentes componentes electrónicos. Por conseguinte, as seguintes ligações podem ser encontradas na área de interface da nossa placa Explorer:

SPI (Serial Perhipheral Interface): Esta ligação é utilizada para a transmissão rápida de dados em série. É normalmente constituída por quatro linhas: MISO (Master In, Slave Out), MOSI (Master Out, Slave In), SCK (Serial Clock) e SS (Slave Select). O SPI é ideal para situações em que é necessária uma elevada taxa de transferência de dados, como no controlo de ecrãs LCD ou cartões SD.

I2C (Inter-Integrated Circuit): I2C é uma interface de dois fios constituída por uma linha de dados (SDA) e uma linha de relógio (SCL). É normalmente utilizada em aplicações de microcontroladores para comunicação entre diferentes circuitos integrados. A sua simplicidade torna-a ideal para aplicações em que não estão disponíveis muitos pinos GPIO.

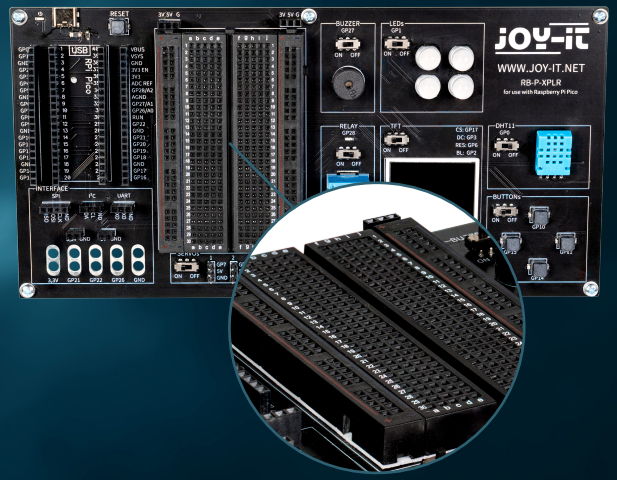
UART (Universal Asynchronous Receiver/Transmitter): Esta interface permite a comunicação em série assíncrona através de duas linhas: TX (Transmissão) e RX (Receção). A UART é frequentemente utilizada para a comunicação entre microcontroladores e computadores ou para ligar módulos como receptores GPS ou módulos Bluetooth.

Ligações de 3,3 V e 5 V: Estas ligações fornecem a fonte de alimentação para os componentes electrónicos. 3,3 V é frequentemente utilizado para microcontroladores e sensores modernos, enquanto 5 V é frequentemente encontrado em dispositivos mais antigos ou que consomem mais energia.

Ligações para clipes de crocodilo: Estes conectores são ideais para ligações temporárias ou para fins de teste. Permitem uma ligação rápida e fácil a vários componentes ou dispositivos de medição sem soldadura. Há um total de cinco conectores deste tipo na Explorer Board, que podem ser utilizados de forma flexível para uma variedade de aplicações.

Cada uma destas ligações tem a sua aplicação e significado específicos na eletrónica, tal como os diferentes tipos de botões numa interface de utilizador têm funções diferentes. Proporcionam a flexibilidade e a funcionalidade necessárias para a criação e expansão de sistemas electrónicos.

4.9 PRANCHETA



As placas de circuito impresso são uma ferramenta indispensável no mundo da eletrónica, tal como os conectores de interface são cruciais para ligar diferentes componentes. Permitem construir e testar circuitos electrónicos rapidamente e sem soldar, o que as torna particularmente populares para prototipagem e fins educativos.

Uma placa de ensaio é normalmente constituída por um bloco de plástico retangular com um grande número de orifícios embutidos dispostos em filas. Estes orifícios estão ligados internamente por traços metálicos que permitem que os componentes e os fios sejam facilmente encaixados e ligados. A disposição normal de uma placa de ensaio inclui duas áreas principais:

Os principais domínios: Estes consistem numa série de filas paralelas de orifícios, normalmente separados por uma ranhura central. Os orifícios de uma fila estão ligados eletricamente uns aos outros. Esta disposição é ideal para inserir circuitos integrados (IC) e outros componentes.

As tomadas eléctricas: Na extremidade da placa de ensaio existem normalmente uma ou duas filas de orifícios que servem de cabos de alimentação. Estas são ligadas verticalmente ao longo de todo o comprimento da placa de ensaio e oferecem uma forma conveniente de fornecer energia e terra em vários pontos do circuito.

A flexibilidade de uma placa de ensaio reside na sua reutilização e na capacidade de construir circuitos sem alterações permanentes. Isto torna-a ideal para a experimentação, uma vez que os erros podem ser facilmente corrigidos e os componentes removidos sem danos. É também uma excelente ferramenta de aprendizagem, uma vez que promove a compreensão da lógica do circuito e das funções dos componentes de uma forma prática e visual.

Além disso, as placas de ensaio estão disponíveis em diferentes tamanhos e com diferentes números de pontos de ligação para satisfazer diferentes necessidades. As placas de ensaio mais pequenas são boas para projectos e experiências simples, enquanto as maiores são adequadas para circuitos mais complexos.

Apesar da sua versatilidade, as placas de ensaio também têm limitações. Não são adequadas para frequências muito elevadas ou para circuitos que exijam elevada potência. Além disso, as ligações podem por vezes ser menos fiáveis do que as ligações soldadas, especialmente se a placa de ensaio se desgastar com o tempo.

Em geral, as placas de ensaio são uma ferramenta essencial para qualquer pessoa que trabalhe com eletrónica - desde principiantes que aprendem o básico até programadores experientes que procuram criar protótipos de forma rápida e eficiente. São o equivalente eletrónico do caderno de esboços de um artista: um local para explorar ideias e experimentar antes de criar o trabalho final.

5. PROJECTOS

Bem-vindo ao capítulo sobre projectos inovadores de eletrónica com o Raspberry Pi Pico! Nesta secção, ser-lhe-á apresentada uma vasta gama de aplicações que vão desde o simples controlo de LEDs até ao desenvolvimento de sistemas mais complexos, como estações meteorológicas automatizadas e sistemas de iluminação dinâmicos. Cada projeto foi cuidadosamente concebido para lhe dar experiência prática com uma variedade de componentes de hardware.

Comece a sua viagem de descoberta com projectos básicos que lhe ensinam a utilizar GPIOs (General Purpose Input/Output) no Raspberry Pi Pico, e aumente as suas competências com tópicos mais avançados, como o controlo de servomotores ou a utilização de sensores para monitorização ambiental. Utilizando componentes como codificadores rotativos, sensores ultra-sónicos, buzzers e LEDs Neopixel, aprenderá a conceber sistemas interactivos e reactivos.

Cada projeto fornece uma introdução detalhada aos componentes necessários, instruções passo a passo sobre a configuração do hardware e exemplos claros de código de programa para o ajudar a compreender os princípios da eletrónica e da programação informática. Também explica como integrar sensores e actuadores externos para recolher e responder a dados em tempo real.

Estes projectos não são apenas educativos, mas também divertidos, com muitas oportunidades de personalização e expansão para que possa desenvolver as suas próprias soluções criativas. Quer seja um principiante que está a começar a explorar o mundo da eletrónica digital ou um programador experiente que procura expandir as suas competências, este capítulo fornece os recursos e a inspiração de que necessita para melhorar as suas competências técnicas e divertir-se a aprender. Prepare-se para expandir as suas competências de programação e eletrónica à medida que é guiado através de cada projeto, enquanto se diverte a criar e a experimentar.

Encontrará os respectivos códigos de exemplo no final de cada projeto. Também pode descarregar os ficheiros **aqui**.

5.1 INDICAÇÃO DA DISTÂNCIA

No nosso primeiro projeto, o nosso objetivo é construir um telémetro ultrassónico que visualize as distâncias no nosso ecrã TFT. Este projeto é uma excelente introdução à deteção e visualização de dados com o Raspberry Pi Pico.

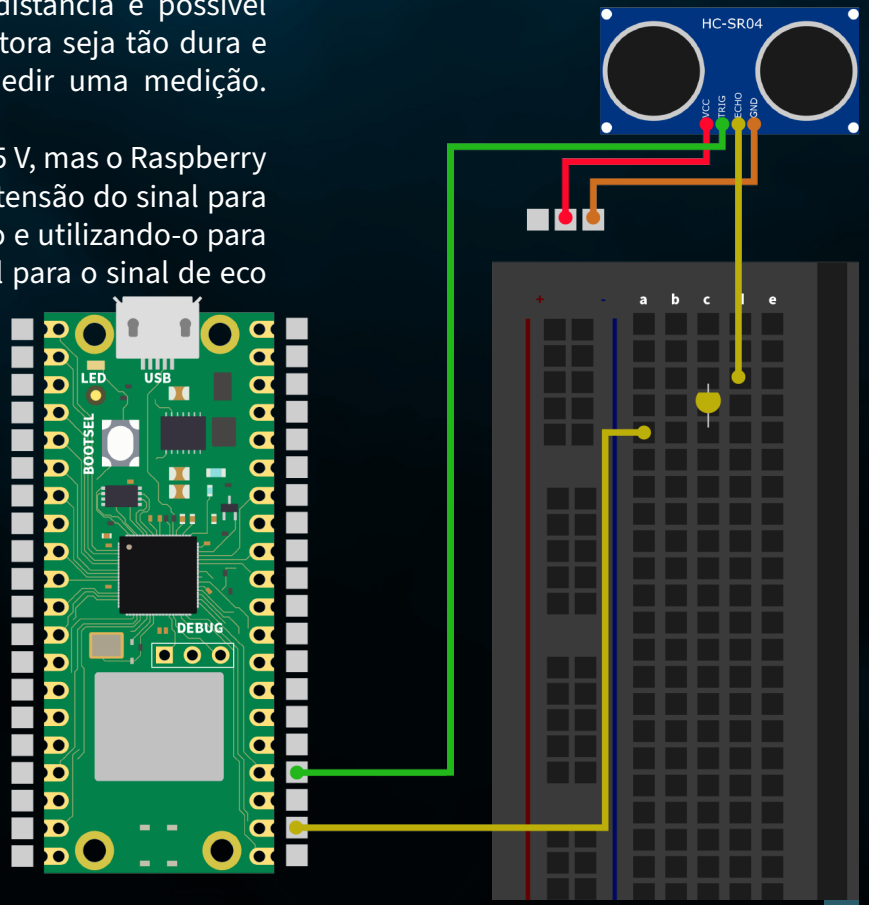
SENSOR ULTRASSÓNICO: Um transmissor emite uma onda ultra-sónica e mede o tempo até que esta seja reflectida e chegue de volta ao transmissor. Uma vez que a velocidade do som é conhecida em vários meios, como o ar (343 m/s a 20 °C) e a água (1.484 m/s), a distância até à superfície reflectora pode ser calculada (reduzindo para metade o tempo de trânsito, uma vez que foi medida a distância de ida e volta). Um impulso do microcontrolador na entrada de disparo desencadeia uma sequência de oito impulsos ultra-sónicos curtos. Assim que o sinal é recebido novamente, a saída de eco fica brevemente alta. O tempo entre o disparo e o sinal de eco corresponde ao tempo de trânsito. A medição da distância é possível na gama de cerca de 2 cm a 400 cm e é bastante precisa, desde que a superfície reflectora seja tão dura e uniforme quanto possível. Materiais macios, tais como alcatifas, podem mesmo impedir uma medição.

Uma vez que o sensor ultrassónico é um sensor que requer uma fonte de alimentação de 5 V, mas o Raspberry Pi Pico só pode processar sinais de 3,3 V sem qualquer problema, é necessário reduzir a tensão do sinal para evitar danos. Conseguimos isto ligando o nosso LED amarelo em série na placa de ensaio e utilizando-o para reduzir a conversão do nosso sinal. O LED também funciona como um indicador de sinal para o sinal de eco ativo.

Mais pormenores sobre os LEDs podem ser encontrados no capítulo 5.5.

RASPBERRY PI PICO	SENSOR ULTRASSÓNICO
3V3	VCC
GP17	Trig
GP16	Echo
GND	GND

ATTENTION! Para este projeto, é necessário colocar os interruptores do relé e do DHT11 em OFF e o interrutor do ecrã TFT em ON.



RESUMO: No nosso primeiro projeto, medimos distâncias com o sensor ultrassónico e visualizamos a distância medida preenchendo o gráfico no ecrã TFT em maior ou menor grau. No nosso exemplo, preenchemos completamente o ecrã a partir de uma distância medida de 100 cm.

```
# Load libraries
from machine import Pin, SPI
import ST7735
import time
import lcd_gfx

# Initialization of GPIO16 as input and GPIO17 as output
trig = Pin(17, Pin.OUT)
echo = Pin(16, Pin.IN, Pin.PULL_DOWN)

# 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)
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)

# Endless loop for measuring the distance
while True:
     # Distance measurement is started using the 10us trigger signal
     trig.value(0)
     time.sleep(0.1)
     trig.value(1)

     # Now wait at the echo input until the signal has been activated
     # Then the time is measured for how long it remains activated
     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
```

Inicialização do sensor ultrassónico e do ecrã TFT

Função auxiliar para ajustar o intervalo de valores

Medição de distâncias

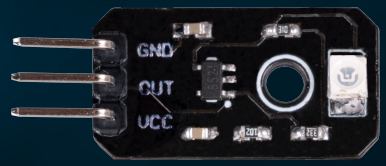
```
# Now the distance is calculated using the recorded time
distance = pulse_duration * 17165 / 1000000
distance = round(distance, 0)

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

# Adjust measured value to LCD height
if(distance > 100):
    distance = 100
drawHeight = round(translate(distance, 0, 100, 0, 160))

# Fill the 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))
```

Cálculo da gama de valores e descrição do ecrã TFT

5.2 ESTAÇÃO METEOROLÓGICA

Mergulhe no segundo projeto da nossa aventura eletrónica, onde criará a sua própria estação meteorológica! Este projeto combina a utilização de um sensor UV com o sensor de temperatura e humidade DHT11 para não só lhe dar uma ideia das condições meteorológicas actuais, mas também medir a radiação UV no local onde se encontra. Toda esta informação é claramente apresentada no ecrã TFT colorido, para que possa ver o tempo e a radiação UV num relance.

SENSOR UV: O sensor UV é um pequeno componente que nos ajuda a medir os raios ultravioleta (UV) invisíveis do sol. Os raios UV são a parte da luz solar que é invisível mas que pode ter um impacto na nossa pele e na nossa saúde. Pense nas queimaduras solares ou no bronzeado da pele - ambos são causados pelos raios UV.

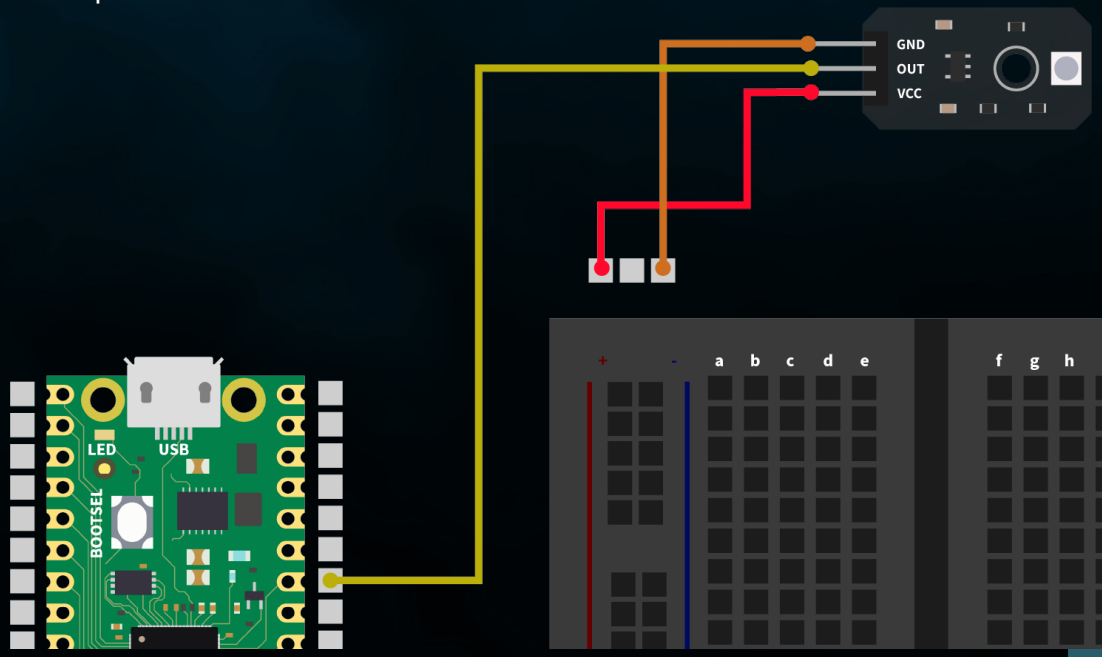


No seu núcleo, o sensor contém um material que reage à radiação UV. Quando os raios UV atingem este material, o sensor altera a sua resistência eléctrica. Esta alteração é convertida pelo sensor num sinal que podemos medir e ler. Com a ajuda do Raspberry Pi Pico, podemos converter este sinal num valor que indica a intensidade atual da radiação UV.

Primeiro, ligue o sensor UV ao seu Raspberry Pi Pico utilizando os cabos fornecidos. Embora também seja possível ligá-lo à placa de ensaio, é muito mais flexível com uma ligação direta por cabo.

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

ATTENTION! Para este projeto, é necessário colocar o interrutor do relé em OFF e os interruptores do ecrã TFT e do sensor DHT11 em ON.



RESUMO: A nossa estação meteorológica lê os sensores DHT11 e UV e apresenta os dados no ecrã TFT.

```
from machine import ADC, Pin, SPI
import utime
import dht
import ST7735  # Assuming this is the library for your TFT display

# Initialize DHT11 sensor
sensor_dht11 = dht.DHT11(Pin(0))

# Initialize UV sensor
uv_sensor = ADC(2)  # Assuming GP28 is ADC pin number 1 in your configuration

# 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=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))

    # Read UV value
    uv_value = uv_sensor.read_u16()
    # Conversion in percent
    uv_percent = (uv_value / 65000) * 100
    print("UV Intensity (percent):", uv_percent)

    # DHT11 Read values
    sensor_dht11.measure()
    temp = sensor_dht11.temperature()
    humid = sensor_dht11.humidity()

    # Display values on LCD
    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))  # Display of UV
intensity in percent with two decimal places

    utime.sleep(10)
```

Inicialização do ecrã TFT

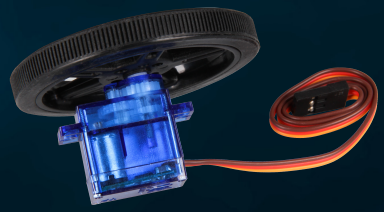
Medição dos valores dos sensores

Saída no ecrã

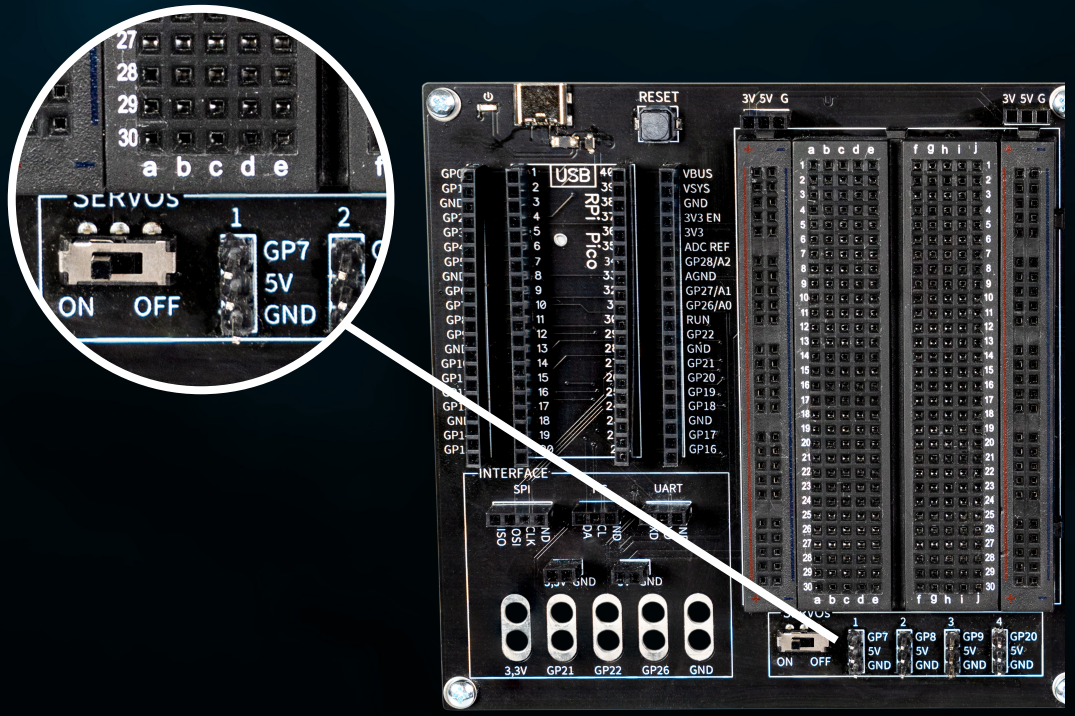
5.3 SERVO CONTROL

Bem-vindo ao terceiro projeto da nossa série de emocionantes aventuras electrónicas com o Explorer Set! Desta vez, trata-se de movimento e controlo. O nosso objetivo é programar e controlar um servomotor de modo a que o seu sentido de rotação possa ser controlado premindo simplesmente um botão. Este projeto não só proporciona uma excelente introdução ao mundo do controlo de motores, como também mostra como reforçar as interacções através de feedback visual num ecrã TFT.

SERVOMOTOR: Um servo é constituído por um motor elétrico com caixa de velocidades e eletrónica de controlo. No lado de saída da caixa de velocidades existe uma roda dentada na qual está montada a roda do servo. Os servos são utilizados na construção de modelos, por exemplo, para controlar a posição da asa ou do leme de um avião ou navio. Cada vez mais servos são também utilizados na engenharia automóvel para fechar automaticamente portas, para reguladores de janelas, espelhos e outros elementos ajustáveis.



Primeiro, ligue o servomotor à interface servo com o número 1 na sua placa Explorer.



ATTENTION! Para este projeto, é necessário colocar o interrutor para o ecrã TFT, os botões e os servos em ON.

RESUMO: Controlamos o nosso servomotor e deixamo-lo alternar entre a rotação esquerda e direita, controlada pelos nossos botões.

Inicialização do servomotor e dos botões

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

# Servo pin numbers
servoOnePin = 7

# Key pin numbers
buttonLeftPin = 15
buttonRightPin = 11

# Initialization of the servo
servoOne = PWM(Pin(servoOnePin))

# Initialize the buttons with PULL_UP to use the default HIGH state
buttonLeft = Pin(buttonLeftPin, Pin.IN, Pin.PULL_UP)
buttonRight = Pin(buttonRightPin, Pin.IN, Pin.PULL_UP)

# Servo speeds in nanoseconds
leftSpeed = 1300000  # Moves the servo to the left
rightSpeed = 1700000  # Moves the servo to the right

# Servo frequency
servoOne.freq(50)  # Typical servo frequency of 50Hz

# Status of the servo
servoState = 'left'  # Starts with counterclockwise rotation

# Last state of the buttons to recognize edges
lastButtonLeft = buttonLeft.value()
lastButtonRight = buttonRight.value()

while True:
    # Read out the current status of the buttons
    currentButtonLeft = buttonLeft.value()
    currentButtonRight = buttonRight.value()

    # Check whether an edge from HIGH to LOW was detected (button was pressed)
    if lastButtonLeft == 1 and currentButtonLeft == 0:
        servoState = 'right'  # Changes the direction to the right when the left
button is pressed
    elif lastButtonRight == 1 and currentButtonRight == 0:
        servoState = 'left'  # Changes the direction to the left when the right
button is pressed
```

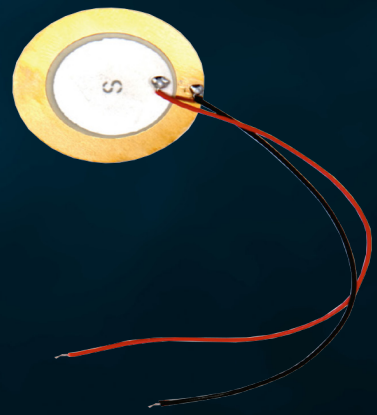
Verificar os botões

```
    # Update the states for the next run
    lastButtonLeft = currentButtonLeft
    lastButtonRight = currentButtonRight

    # Controls the servo based on the current status
    if servoState == 'left':
        servoOne.duty_ns(leftSpeed)  # Moves the servo to the left
    else:
        servoOne.duty_ns(rightSpeed)  # Moves the servo to the right

    sleep(0.1)  # Short break for the control cycle
```

Servo control

5.4 CAMPAINHA DE FABRICO PRÓPRIO

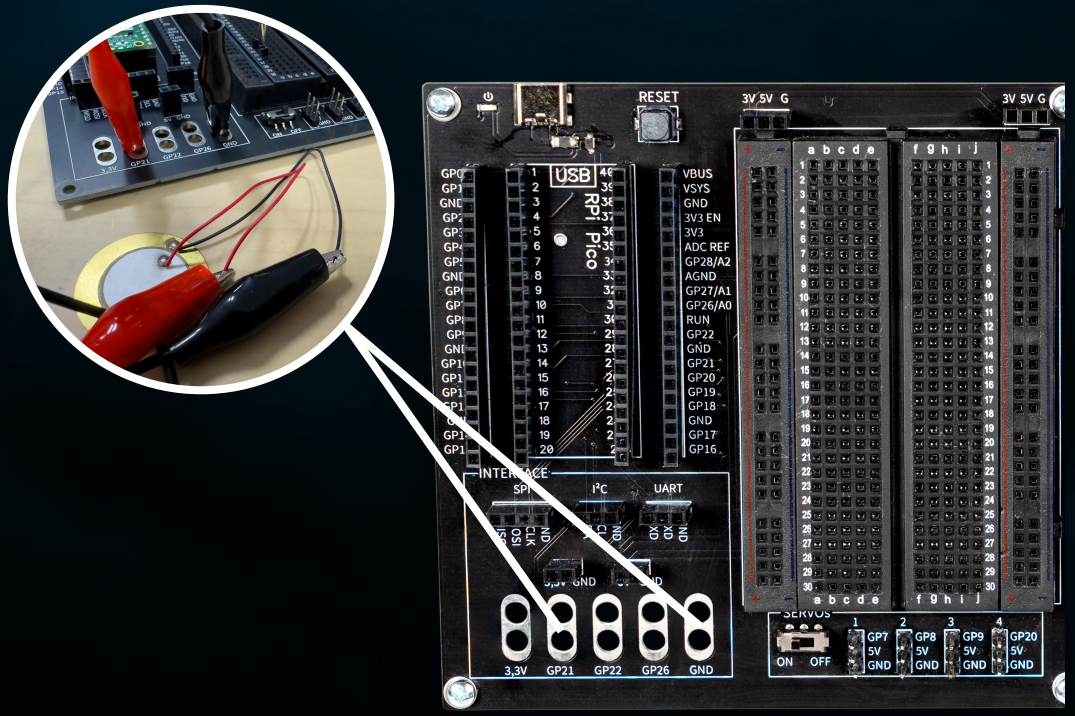
O quarto projeto da nossa aventura eletrónica com o Explorer Set tem tudo a ver com o som! Mergulhamos no mundo dos sinais acústicos, criando o nosso próprio circuito de campainha. Este projeto não só lhe dá a oportunidade de compreender os fundamentos da criação de circuitos, mas também de aprender a criar sinais sonoros utilizando ferramentas simples. A utilização de pinças de crocodilo torna a construção particularmente fácil e acessível, mesmo para aqueles que ainda estão no início da sua jornada eletrónica.



Em primeiro lugar, ligamos cuidadosamente a campainha à Explorer Board utilizando clipes de crocodilo. Estes terminais são ideais para ligações rápidas e flexíveis, sem necessidade de soldar. O buzzer, um pequeno componente capaz de produzir som, torna-se a peça central do nosso projeto. Quando a corrente é aplicada, a campainha vibra e produz um som.

Primeiro, liga um clipe de crocodilo ao conetor de clipe de crocodilo GP21 na tua placa Explorer. Liga a outra extremidade do clip crocodilo ao cabo vermelho do buzzer. Liga outro clipe de crocodilo à ligação de clipe de crocodilo GND na tua placa Explorer. Ligar a outra extremidade do terminal ao cabo preto do sinal sonoro.

RASPBERRY PI PICO	BUZINA
GP21	Cabo vermelho
GND	Cabo preto



ATTENTION! Para este projeto, é necessário colocar o interrutor dos botões em ON.

RESUMO: Ligamos um sinal sonoro externo ao nosso Raspberry Pi Pico para tocar uma música simples e divertida.

```
from machine import Pin, PWM
import utime
# Note frequencies (in Hz)
notes = {
    'C4': 262,
    'D4': 294,
    'E4': 330,
    'F4': 349,
    'G4': 392,
    'A4': 440,
    'B4': 494,
    'C5': 523
}
# Melody and duration (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]
# Initialization of the buzzer
buzzer = PWM(Pin(21))
buzzer.freq(440) # Set a start frequency
# Function to play a note
def play_note(note, duration):
    if note in notes:
        buzzer.freq(notes[note]) # Set the frequency based on the note
        buzzer.duty_u16(32767) # Start the PWM signal
        utime.sleep_ms(duration) # Hold the note for the duration
    buzzer.duty_u16(0) # Stop the PWM signal (switch off note)
    utime.sleep_ms(50) # Short pause between the notes
# Play the melody
for note, duration in zip(melody, durations):
    play_note(note, duration)
buzzer.deinit() # Deactivate the PWM channel when finished
```

Lista de notas

Memória melódica

Função para tocar as notas

5.5 O SEU PRÓPRIO CIRCUITO

No quinto projeto da nossa aventura eletrónica com o Explorer Set, exploramos o fascinante mundo do controlo da luz. Desta vez, estamos a construir um circuito que pode ser utilizado para controlar LEDs. Esta é uma oportunidade fantástica para compreender os princípios dos circuitos electrónicos e, ao mesmo tempo, controlar os efeitos de iluminação.

LEDS: Os LEDs, ou díodos emissores de luz, são fontes de luz pequenas mas potentes que são utilizadas em muitos projectos electrónicos. Têm muitas vantagens em relação às lâmpadas tradicionais, tais como uma vida útil mais longa, um menor consumo de energia e a capacidade de se iluminarem com cores diferentes. Um LED é constituído por um material semicondutor que emite luz quando é percorrido por uma corrente eléctrica.

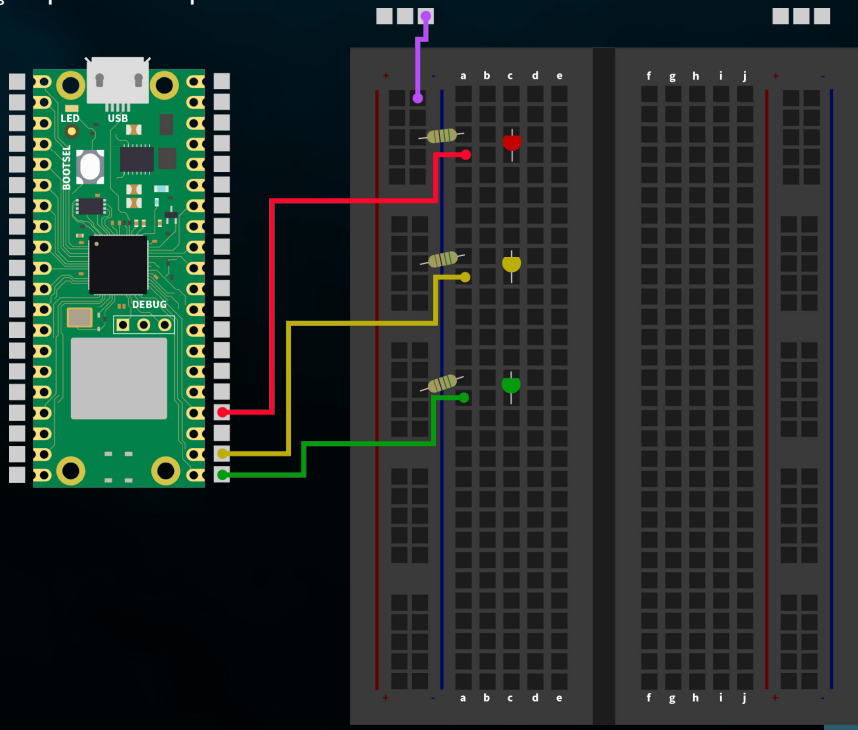
É importante reconhecer a polaridade correcta dos LEDs, uma vez que estes só funcionam se a corrente fluir através deles na direção certa. Isto significa que o pólo positivo da fonte de alimentação deve ser ligado à extremidade positiva do LED e o pólo negativo da fonte de alimentação deve ser ligado à extremidade negativa do LED.

É assim que se reconhece a polaridade de um LED:

Perna mais longa: para a maioria dos LEDs, a perna mais longa é o ânodo (+), ou seja, a ligação positiva. A perna mais curta é o cátodo (-), ou seja, a ligação negativa.

Aresta plana: Pode haver uma borda plana na lateral do invólucro do LED. Este lado marca normalmente o cátodo, ou seja, o pólo negativo.

Primeiro, reconstrua o circuito como mostra o diagrama seguinte. Mas certifique-se de que a polaridade dos LEDs está correcta. Utilize também uma resistência em série de 56 Ω (verde-azul-preto) para cada LED.



RASPBERRY PI PICO	LEDS
GP18	LED vermelho
GP17	LED amarelo
GP16	LED verde

ATTENTION! Para este projeto, é necessário colocar o interrutor do ecrã TFT em OFF.

RESUMO: Controlamos três LEDs diferentes através dos pinos do Raspberry Pi Pico, com cada LED a piscar alternadamente.

Inicialização dos LEDs

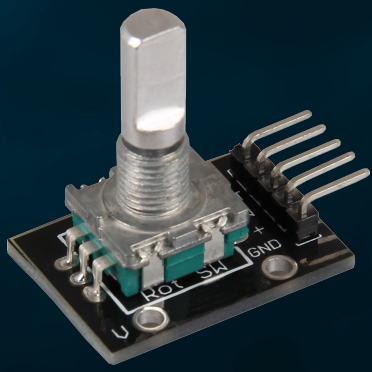
```python
from machine import Pin
import utime

# Initialize the LEDs
red_led = Pin(18, Pin.OUT)
yellow_led = Pin(17, Pin.OUT)
green_led = Pin(16, Pin.OUT)

# Flashing function for one LED
def blink_led(led, duration):
    led.value(1)  # Switch on LED
    utime.sleep(duration)
    led.value(0)  # Switch off LED
    utime.sleep(duration)

# Hauptschleife
while True:
    blink_led(red_led, 0.5)     # Red LED flashes for 0.5 seconds
    blink_led(yellow_led, 0.5) # Yellow LED flashes for 0.5 seconds
    blink_led(green_led, 0.5)   # Green LED flashes for 0.5 seconds
```

Função de intermitência do LED

Circuito principal

5.6 CONTROLO LED

No sexto projeto da nossa aventura eletrónica, utilizamos o nosso codificador rotativo para controlar o brilho e a cor dos LEDs - uma forma simples mas fascinante de mergulhar na eletrónica. O codificador rotativo, o nosso elemento central de controlo, permite uma interação lúdica graças à sua dupla função. As alterações de luminosidade são controladas por rotação, ao passo que, ao premir o codificador, as cores dos LEDs são alternadas - ideal para ilustrar os princípios básicos da eletrónica e da mistura de cores.



CODIFICADOR ROTATIVO: O codificador rotativo é um pequeno dispositivo inteligente que converte os seus movimentos rotativos em sinais electrónicos. Imagine um botão rotativo como o que conhece de um rádio. Quando roda este botão, o codificador rotativo pode medir a distância e a direção em que o rodou. Esta informação pode então ser utilizada, por exemplo, para alterar o volume, navegar nos menus ou, nos nossos projectos, ajustar o brilho dos LEDs.

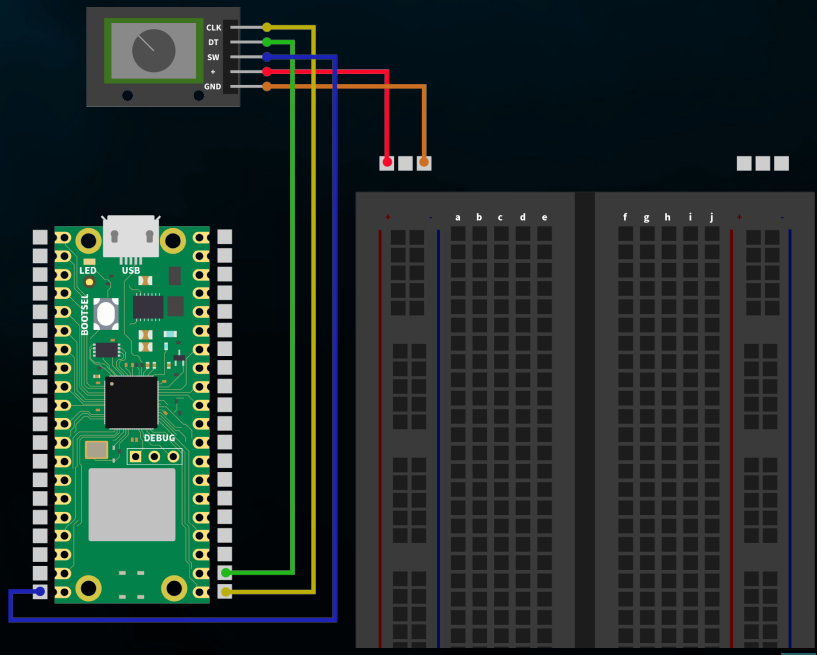
Os codificadores rotativos também têm frequentemente um botão incorporado - isto significa que também podem funcionar como um interrutor de pressão. Quando se pressiona o botão rotativo, o codificador rotativo reconhece esta pressão como um sinal separado. Isto pode ser utilizado para várias funções, como ligar e desligar um dispositivo ou alterar os modos de funcionamento.

PARA RESUMIR: Um codificador rotativo permite-lhe enviar vários comandos para os seus projectos electrónicos rodando e pressionando. É uma ferramenta intuitiva e versátil que torna as interacções com os seus projectos fáceis e divertidas.

Primeiro, ligue o codificador rotativo ao seu Raspberry Pi Pico da seguinte forma:

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

ATTENTION! Para este projeto, é necessário colocar o interrutor do ecrã TFT em OFF e o interrutor dos LEDs em ON.



RESUMO: Utilizamos o codificador rotativo para controlar a cor e o brilho dos nossos quatro LEDs. Rodar o codificador altera o brilho, enquanto premir o codificador ajusta a cor dos LEDs.

Inicialização dos LEDs e do codificador rotativo

```
from machine import Pin
import utime
import neopixel

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

# Rotate encoder 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)

# Global variables
counter = 0
PIN_CLK_LAST = PIN_CLK.value()
delayTime = 0.001
debounce_time_encoder = 0
debounce_time_button = 0

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

# Initialize brightness
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()
```

Função do codificador rotativo

```
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)
```

Lista de notas

5.7 CONTROLO AUTOMÁTICO DA LUMINOSIDADE

No sétimo projeto da nossa aventura eletrónica, utilizamos um fotodíodo para controlar automaticamente o brilho dos LEDs. O fotodíodo converte a luz num sinal elétrico, de modo que os LEDs ficam mais brilhantes quando está escuro e mais fracos quando há mais luz ambiente.

Ligando o fotodíodo à Explorer Board e programando-o no Raspberry Pi Pico, os LEDs adaptam-se de forma inteligente à luminosidade ambiente. Este projeto mostra como é possível construir sistemas electrónicos reactivos e economizadores de energia com componentes simples.

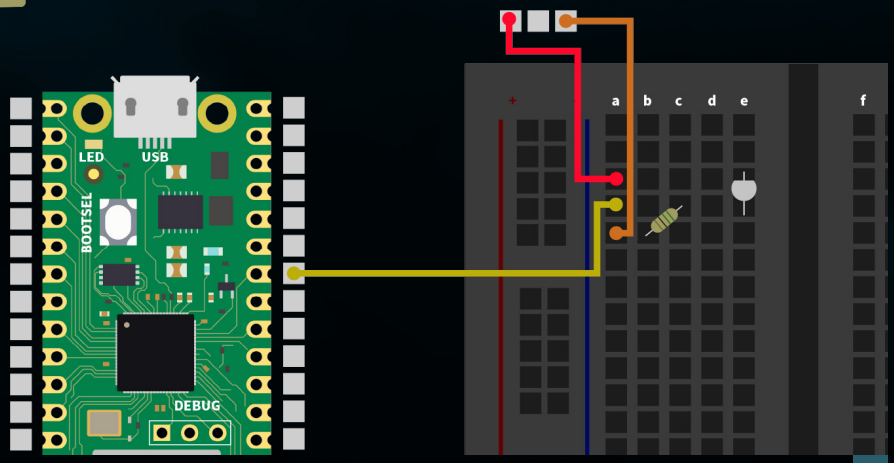
FOTODÍODO: Um fotodíodo é um tipo especial de semicondutor que responde à luz que o atinge gerando uma corrente eléctrica. Pense num fotodíodo como um pequeno painel solar: quando a luz incide sobre ele, converte essa luz num sinal elétrico. Quanto mais luz atingir o fotodíodo, mais forte se torna o sinal.

Os fotodíodos são muito sensíveis e podem detetar mesmo pequenas quantidades de luz, o que os torna ideais para projectos em que é importante medir o brilho ou a presença de luz. Por exemplo, podem ser utilizados em controladores automáticos de luminosidade, sensores de luz ou como parte de um sistema de controlo de iluminação.

Em suma, os fotodíodos são detectores de luz eficazes que nos permitem fazer com que os dispositivos electrónicos reajam de forma inteligente às alterações da iluminação ambiente.

Em primeiro lugar, ligue o fotodíodo através da placa de ensaio da seguinte forma. Tenha em atenção que a utilização de uma resistência também é necessária aqui. Utilize aqui a resistência de 100 kΩ (castanho-preto-amarelo).

RASPBERRY PI PICO	FOTODÍODO
3V3	Cátodo positivo
GP28/A2	Ânodo negativo



ATTENTION! Para este projeto, é necessário colocar o interrutor para o relé em OFF e o interrutor para os LEDs em ON.

RESUMO: Utilizamos o nosso fotodíodo para medir a luminosidade ambiente e ajustar a luminosidade de quatro LEDs. A intensidade dos LEDs muda de acordo com a luz detectada pelo fotodíodo, sendo que ambientes mais escuros levam a LEDs mais brilhantes e vice-versa. É preferível utilizar uma lanterna para obter o melhor resultado possível.

```
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)

# Photodiode setup on ADC pin GP28 (A2)
fotodiode = ADC(2)

# Conversion function for brightness values of the photodiode into a suitable
brightness for the LEDs
def brightness_from_light(sensor_value):
    # Minimum and maximum sensor value
    min_sensor_value = 400
    max_sensor_value = 10000

    # Invert the sensor value within the actual range
    normalized_value = max_sensor_value - sensor_value + min_sensor_value

    # Scale the inverted value to a brightness range (0.05 to 0.5)
    # Adjust the scaling: Divide by (max_sensor_value - min_sensor_value)
    return max(0.05, min(0.5, normalized_value / (max_sensor_value - min_sensor_
value) * 0.45 + 0.05))

def update_leds(brightness):
    color = (255, 255, 255)  # White
    dimmed_color = tuple([int(c * brightness) for c in color])
    for i in range(NUM_LEDS):
        np[i] = dimmed_color
    np.write()


while True:
    # Read the sensor value from the photodiode
    light_value = fotodiode.read_u16()
    print(light_value)

    # Calculate the brightness based on the sensor value
    brightness = brightness_from_light(light_value)

    # Update the LEDs with the new brightness
    update_leds(brightness)

    # Waiting time to reduce the load on the CPU and for smoother brightness
transitions
    utime.sleep(0.5)
```

Inicialização dos LEDs e do fotodíodo

Medição do fotodíodo e controlo da luminosidade

5.8 CONTROLO DE LED RGB

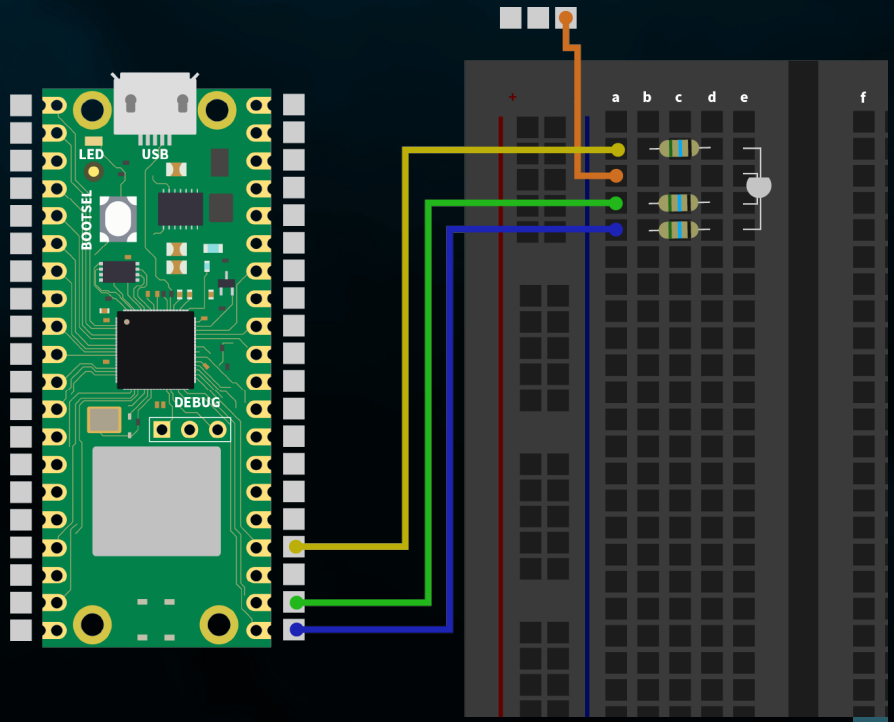
No oitavo e último projeto da nossa série de eletrónica, vamos concentrar-nos no controlo de cores dos LED RGB utilizando os botões integrados na Explorer Board. Os LEDs RGB são díodos emissores de luz especiais que combinam luz vermelha, verde e azul (RGB) para apresentar uma vasta gama de cores. Ao ajustar a intensidade de cada componente de cor individualmente, podemos criar praticamente qualquer cor.

Neste projeto, ligamos o LED RGB à placa de ensaio e utilizamos os botões existentes para controlar as cores do LED. Cada botão é atribuído a uma cor (vermelho, verde, azul).

LED RGB: Um LED RGB combina vermelho, verde e azul num único ponto de luz. Ao alterar o brilho de cada uma das três cores, é possível criar praticamente qualquer cor. Isto é feito por modulação de largura de pulso (PWM), que controla a intensidade de cada cor. Assim, os LED RGB com apenas três cores permitem um amplo espetro de cores, ideal para projectos de iluminação coloridos.

Primeiro, ligue o LED RGB à placa de ensaio da seguinte forma. Observe que cada um dos três canais de cores também requer um resistor em série aqui. Deve utilizar aqui a resistência de 56 Ω (verde-azul-preto).

RASPBERRY PI PICO	RGB-LED
GP18	Primeiro pino
GND	Segundo pino
GP17	Terceiro pino
GP16	Quarto pino



ATTENTION! Para este projeto, é necessário colocar o interrutor do TFT em OFF e o dos botões em ON.

RESUMO: Os três canais de cor do LED RGB (vermelho, verde e azul) são activados e desactivados com os botões (esquerdo, superior e direito).

```
from machine import Pin
import utime

# Initialize the LED pins
red_led = Pin(18, Pin.OUT)
green_led = Pin(17, Pin.OUT)
blue_led = Pin(16, Pin.OUT)

# Initialize the 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)

# Save states of the LEDs
red_state = False
green_state = False
blue_state = False

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

while True:
    # Check the status of the red button
    if button_red.value() == 0:
        red_state = not red_state
        toggle_led(red_led, red_state)
        utime.sleep(0.2)  # Entprellung

    # Check the status of the green button
    if button_green.value() == 0:
        green_state = not green_state
        toggle_led(green_led, green_state)
        utime.sleep(0.2)  # Entprellung

    # Check the status of the blue button
    if button_blue.value() == 0:
        blue_state = not blue_state
        toggle_led(blue_led, blue_state)
        utime.sleep(0.2)  # Debouncing
```

Inicialização do LED e dos botões

Testar os botões e controlar o LED

6. OBRIGAÇÕES DE INFORMAÇÃO E DE RETOMA

AS NOSSAS OBRIGAÇÕES DE INFORMAÇÃO E DE RETOMA AO ABRIGO DA LEI ALEMÃ RELATIVA AOS EQUIPAMENTOS ELÉCTRICOS E ELECTRÓNICOS (ELEKTROG)

SÍMBOLO NOS EQUIPAMENTOS ELÉCTRICOS E ELECTRÓNICOS: 
Este caixote do lixo riscado significa que os aparelhos eléctricos e electrónicos não devem ser colocados no lixo doméstico. Deve entregar os aparelhos velhos num ponto de recolha. Antes de os entregar, deve separar as pilhas e acumuladores usados que não estejam incluídos no aparelho antigo.

OPÇÕES DE DEVOLUÇÃO:
Enquanto utilizador final, pode devolver o seu aparelho antigo (que desempenha essencialmente a mesma função que o aparelho novo que nos foi comprado) para eliminação, sem custos, quando adquire um aparelho novo. Os pequenos electrodomésticos sem dimensões exteriores superiores a 25 cm podem ser eliminados em quantidades domésticas normais, independentemente de ter adquirido um novo aparelho.

POSSIBILIDADE DE DEVOLUÇÃO NAS INSTALAÇÕES DA NOSSA EMPRESA DURANTE O HORÁRIO DE FUNCIONAMENTO:
SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

OPÇÃO DE DEVOLUÇÃO NA SUA ÁREA:
Enviar-lhe-emos um selo de encomenda com o qual poderá devolver-nos o aparelho gratuitamente. Para tal, contacte-nos por correio eletrónico para service@joy-it.net ou por telefone.

INFORMAÇÕES SOBRE A EMBALAGEM:
Embale o seu aparelho usado de forma segura para o transporte. Se não dispuser de material de embalagem adequado ou não quiser utilizar o seu próprio material, contacte-nos e enviar-lhe-emos a embalagem adequada.

7. APOIO

Também estamos ao seu dispor após a compra. Se alguma pergunta ficar por responder ou se surgirem problemas, estamos também disponíveis por correio eletrónico, telefone e sistema de apoio por bilhete.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Telefone: +49 (0)2845 9360 – 50 (Seg. - Qui.: 09:00 - 17:00 ou relógio, Sex: 09:00 - 14:30 ou relógio)

Para mais informações, visite o nosso sítio Web:

WWW.JOY-IT.NET