EXPLORER SET

RB-P-XPLR-SET

TABLE DES MATIÈRES

1. INFORMATIONS GÉNÉRALES

Cher client, nous vous remercions d'avoir choisi notre produit. Dans ce qui suit, nous allons vous montrer ce dont vous devez tenir compte lors de la mise en service et de l'utilisation.

Si vous rencontrez des problèmes inattendus lors de l'utilisation, n'hésitez pas à nous contacter.

2. VUE D'ENSEMBLE DE L'APPAREIL ET AFFECTATION DES BROCHES

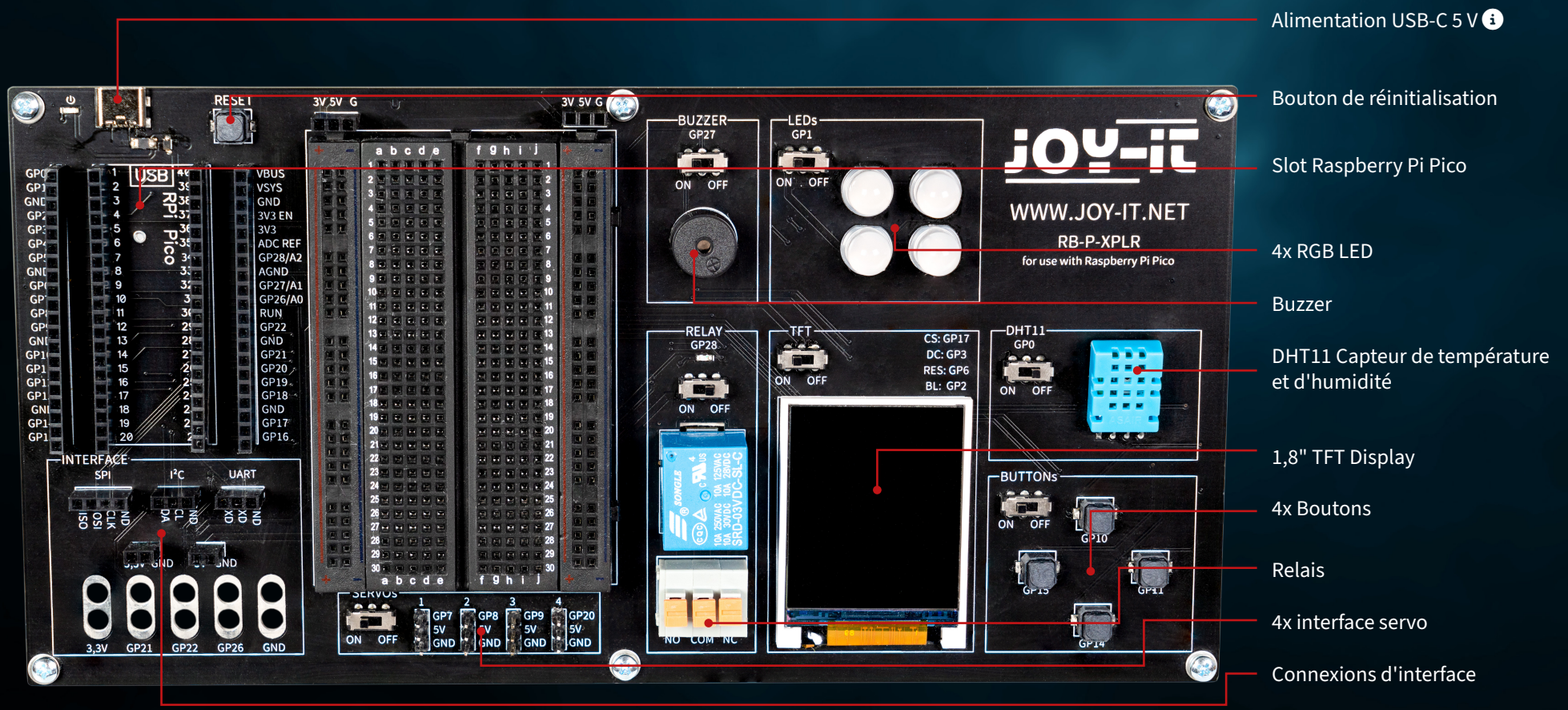
Notre carte Explorer est un moyen simple et efficace de développer vos projets Raspberry Pi Pico.

Les composants les plus importants étant déjà intégrés, vous gagnez du temps et de l'énergie lors du câblage. La carte Explorer dispose d'une large gamme de connecteurs d'interface afin que vous puissiez connecter vos projets à une variété de modules et d'appareils. Grâce à la planche à pain intégrée, vous pouvez rapidement construire et réaliser vos propres projets.

Grâce à la possibilité d'activer ou de désactiver tous les modules individuellement, vous pouvez à tout moment utiliser vos broches, qui sont également acheminées séparément vers l'extérieur, pour d'autres projets ou expérimenter sur la planche à pain intégrée.

Tous les composants intégrés peuvent être désactivés via l'interrupteur correspondant s'ils ne sont pas nécessaires. Cela signifie que les broches associées peuvent également être utilisées pour d'autres composants si nécessaire.

À gauche et à droite du Raspberry Pi Pico, toutes les broches sont conçues de manière supplémentaire. Les composants peuvent y être connectés directement ou acheminés vers la carte à pain intégrée via des câbles supplémentaires.

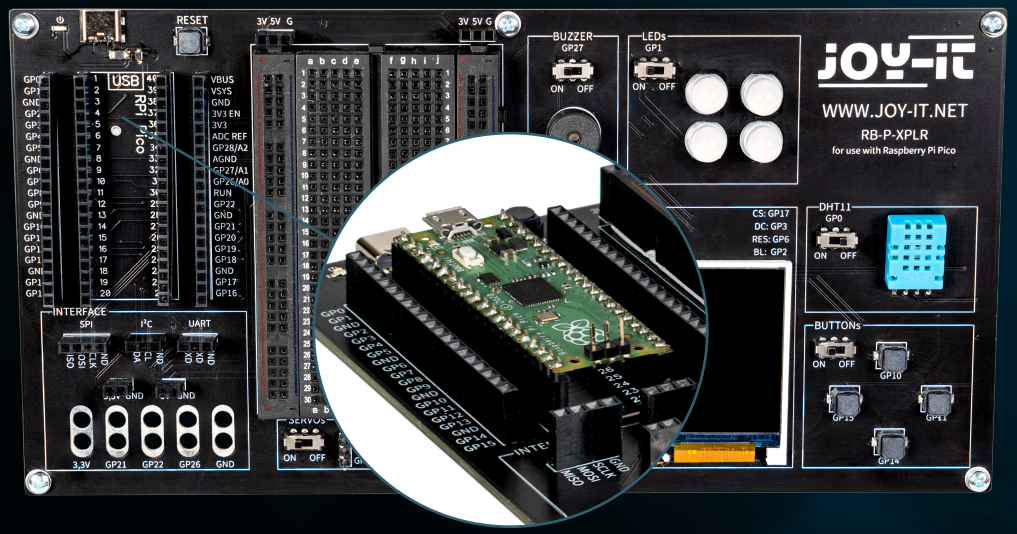


Veuillez noter que la connexion USB-C doit toujours être connectée pour être utilisée. Une alimentation via la connexion micro USB du Raspberry Pi Pico n'est pas possible.

AFFECTATION DES BROCHES	
Buzzer	GP27
LEDs	GP1
Relais	GP28
1,8" TFT Display	CS: GP17, DC: GP3, RES: GP6, BL: GP2
DHT11	GP0
Boutons	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

Branchez d'abord votre Raspberry Pi Pico dans la fente de votre carte.



Connectez maintenant un câble micro USB à votre ordinateur et au Raspberry Pi Pico pour la programmation.

ATTENTION! Le port USB-C de la carte Explorer est utilisé exclusivement pour l'alimentation électrique. Il n'est pas utilisé pour transférer des données vers le Raspberry Pi. Vous pouvez utiliser un programme de développement approprié de votre choix pour transférer notre programme d'exemple. Nous recommandons **Thonny Python IDE**.

ATTENTION! Si vous êtes nouveau dans le monde des microcontrôleurs et de l'électronique, ne vous inquiétez pas ! Nous avons préparé un guide spécial pour les débutants. Ce guide est spécialement adapté aux besoins des débutants et explique comment utiliser le Raspberry Pi Pico étape par étape.

De la configuration de base à l'exécution de projets, ce guide vous accompagne tout au long du processus. Notre guide comprend des explications faciles à comprendre et des conseils utiles pour vous aider à développer rapidement et efficacement vos compétences à grande échelle avec le Raspberry Pi Pico. Vous pouvez télécharger notre **guide ici**.

4. LES MODULES EN DÉTAIL

Dans ce qui suit, tous les modules disponibles sur la carte Explorer sont expliqués individuellement avec des exemples de code. Vous pouvez télécharger ici tous les exemples de codes et de bibliothèques, ainsi qu'un exemple de code qui relie tous les modules entre eux.

Pour l'utilisation de certains modules, des bibliothèques externes et un fichier de police sont utilisés. Téléchargez les bibliothèques et chargez-les dans le dossier lib de votre Raspberry Pi Pico. Placez le fichier de police dans le répertoire racine de votre Raspberry Pi Pico.

4.1 BUZZER

Un buzzer produit un signal sonore, comme un haut-parleur. Toutefois, contrairement à un haut-parleur, il ne convient qu'à une gamme de fréquences limitée, de sorte qu'il ne produit pas un bon son pour reproduire de la musique ou de la parole. En revanche, il est idéal pour générer des signaux d'avertissement puissants sous la forme de bips. Chaque fois qu'un appareil électrique génère un signal sonore, il s'agit presque toujours d'un buzzer. C'est le cas, par exemple, des réveils, des détecteurs de fumée ou du rappel de bouclage des ceintures de sécurité dans les voitures.

Le buzzer est connecté à la broche 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 RGB LEDS

Les LED RVB sont un type de diode électroluminescente qui combine le rouge, le vert et le bleu pour produire une variété de couleurs. Tout comme un buzzer ne produit que des sons simples, les LED RVB ne peuvent pas afficher d'images complexes, mais elles sont excellentes pour mélanger et varier les couleurs. Chaque LED d'une unité RVB peut varier en intensité pour produire différentes teintes, des pastels doux aux couleurs vives et saturées. Elles sont donc idéales pour l'éclairage d'ambiance, l'éclairage décoratif et dans les applications où des signaux visuels sont nécessaires, comme dans les configurations de jeu ou comme indicateurs d'état dans les appareils électroniques. Leur polyvalence et leur efficacité énergétique en ont fait un choix populaire dans les systèmes d'éclairage modernes, bien que, comme le buzzer, leur fonctionnement simple signifie qu'ils ne peuvent pas créer d'images ou de motifs complexes sans unités de contrôle supplémentaires.

Les DEL GPIO sont connectées à la broche 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 RELAIS

Les relais comptent parmi les plus anciens composants électromécaniques et fonctionnent comme des interrupteurs à commande électrique. Avec une faible tension d'entrée et un faible courant, une charge électrique importante peut être activée et désactivée à la sortie. Lorsque le relais commute, la LED rouge s'allume également. Vous pouvez insérer des extrémités de câble dénudées dans la prise de raccordement (en appuyant sur le levier orange) pour utiliser les trois connexions.

Le relais est connecté à la broche 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

L'écran à cristaux liquides (LCD TFT) d'environ 65 000 couleurs et d'une diagonale de 1,8 pouce a une résolution de 128×160 pixels et peut être contrôlé via SPI. Il convient à l'affichage de graphiques et d'images colorés. Les lettres et autres caractères sont affichés sous forme de graphiques composés de nombreux points individuels.

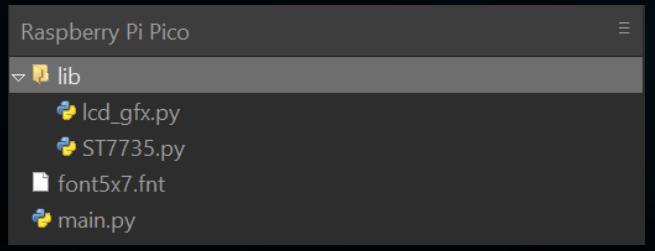
Le TFT est connecté aux broches GPIO GP17 (CS), GP3 (DC), GP6 (RES) et GP2 (BL).

```
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!‘)
```

Outre les textes, des rectangles peuvent également être affichés, par exemple :

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

ATTENTION! Deux fichiers de bibliothèque distincts et un fichier de police sont nécessaires pour l'écran TFT ; vous pouvez télécharger les fichiers requis ici. Transférez ensuite tous les fichiers du dossier Libraries dans le répertoire racine de votre Raspberry Pi Pico de manière à ce que la structure du dossier ressemble à ceci :



4.5 DHT 11

Le capteur DHT11 peut détecter des températures de 0 °C à 50 °C (précision de ±2 °C) et une humidité relative de 20 % à 80 % (±5 %) (au maximum une fois par seconde). Les stations météorologiques constituent probablement le principal domaine d'application d'un capteur tel que le DHT11. Pour tester la fonctionnalité, il suffit d'approcher la bouche du capteur et d'expirer lentement. L'air respiré diffère de l'environnement en termes de température et d'humidité, ce qui devrait entraîner une modification significative des valeurs.

Le DHT11 est connecté à la broche 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 BOUTONS

Les boutons sont des éléments interactifs des interfaces utilisateur qui remplissent une fonction simple mais essentielle : la saisie par l'utilisateur. Tout comme les LED RVB peuvent afficher une variété de couleurs, les boutons sont utilisés pour lancer un large éventail de commandes et d'actions dans les environnements numériques.

Les boutons sont connectés aux broches GPIO GP10 (en haut), GP11 (à droite), GP14 (en bas) et GP15 (à gauche).

```
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

Un servo se compose d'un moteur électrique, d'un réducteur et d'une électronique de commande. Du côté de la sortie du réducteur se trouve une roue dentée sur laquelle est montée la corne du servo. Les servos sont utilisés dans le modélisme, par exemple pour contrôler la position des ailes ou du gouvernail d'un avion ou d'un bateau. De plus en plus de servos sont également utilisés dans la construction automobile pour fermer automatiquement les portes, pour les lève-vitres, les rétroviseurs et d'autres éléments réglables.

Les connexions servo sont les broches GPIO GP7, GP8, GP9 et 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

Les connexions d'interface jouent un rôle crucial dans le monde de l'électronique, à l'instar des boutons dans les interfaces utilisateur. Elles permettent la communication et l'alimentation électrique entre différents composants électroniques. Les connexions suivantes se trouvent donc dans la zone d'interface de notre Explorer Board :

SPI (Serial Perhipheral Interface): Cette connexion est utilisée pour la transmission rapide de données en série. Elle se compose généralement de quatre lignes : MISO (Master In, Slave Out), MOSI (Master Out, Slave In), SCK (Serial Clock) et SS (Slave Select). SPI est idéal pour les situations où un taux de transfert de données élevé est nécessaire, par exemple lors du contrôle d'écrans LCD ou de cartes SD.

I2C (Inter-Integrated Circuit): I2C est une interface bifilaire composée d'une ligne de données (SDA) et d'une ligne d'horloge (SCL). Elle est couramment utilisée dans les applications de microcontrôleurs pour la communication entre différents circuits intégrés. Sa simplicité la rend idéale pour les applications où il n'y a pas beaucoup de broches GPIO disponibles.

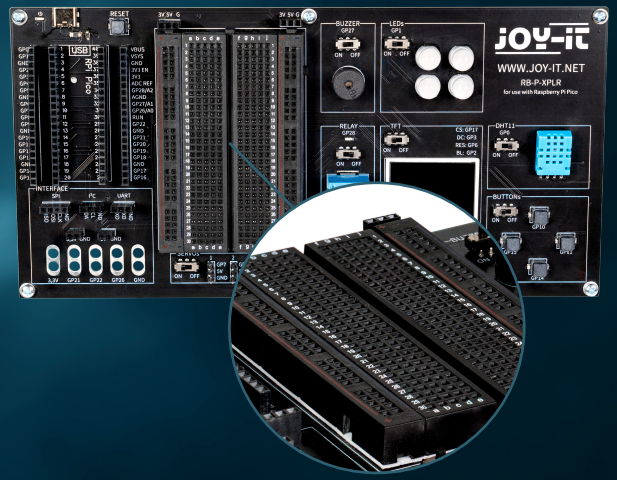
UART (Universal Asynchronous Receiver/Transmitter): Cette interface permet une communication série asynchrone via deux lignes : TX (Transmit) et RX (Receive). L'UART est souvent utilisée pour la communication entre les microcontrôleurs et les ordinateurs ou pour la connexion de modules tels que les récepteurs GPS ou les modules Bluetooth.

Connexions 3,3 V et 5 V: Ces connexions permettent d'alimenter les composants électroniques. La tension de 3,3 V est souvent utilisée pour les microcontrôleurs et les capteurs modernes, tandis que la tension de 5 V est souvent utilisée pour les appareils plus anciens ou plus gourmands en énergie.

Connexions pour pinces crocodiles: Ces connecteurs sont idéaux pour les connexions temporaires ou à des fins de test. Ils permettent une connexion rapide et facile à divers composants ou appareils de mesure sans soudure. La carte Explorer comporte au total cinq connecteurs de ce type, qui peuvent être utilisés de manière flexible pour toute une série d'applications.

Chacune de ces connexions a une application et une signification spécifiques en électronique, de la même manière que les différents types de boutons d'une interface utilisateur ont des fonctions différentes. Elles offrent la flexibilité et la fonctionnalité nécessaires à la mise en place et à l'extension des systèmes électroniques.

4.9 PLANCHE À PAIN



Les planches à pain sont un outil indispensable dans le monde de l'électronique, tout comme les connecteurs d'interface sont cruciaux pour relier différents composants. Elles permettent de construire et de tester des circuits électroniques rapidement et sans soudure, ce qui les rend particulièrement populaires pour le prototypage et à des fins éducatives.

Une planche à pain se compose généralement d'un bloc de plastique rectangulaire comportant un grand nombre de trous encastrés disposés en rangées. Ces trous sont reliés à l'intérieur par des traces métalliques qui permettent de brancher et de connecter facilement les composants et les fils. La disposition standard d'une planche à pain comprend deux zones principales :

Les principaux domaines : Ils consistent en une série de rangées parallèles de trous, généralement séparées par une rainure centrale. Les trous d'une même rangée sont reliés électriquement les uns aux autres. Cette disposition est idéale pour insérer des circuits intégrés (CI) et d'autres composants.

Les multiprises : Sur le bord de la plaque, il y a généralement une ou deux rangées de trous qui servent de barrettes d'alimentation. Celles-ci sont connectées verticalement sur toute la longueur de la planche à pain et offrent un moyen pratique de fournir l'alimentation et la mise à la terre en divers points du circuit.

La flexibilité d'une planche à pain réside dans sa réutilisation et dans la possibilité de construire des circuits sans modifications permanentes. Elle est donc idéale pour l'expérimentation, car les erreurs peuvent être facilement corrigées et les composants retirés sans dommage. Il s'agit également d'un excellent outil d'apprentissage, car il permet de comprendre la logique des circuits et les fonctions des composants d'une manière pratique et visuelle.

En outre, les planches à pain sont disponibles en différentes tailles et avec différents nombres de points de connexion pour répondre à différents besoins. Les petites planches à pain conviennent aux projets et expériences simples, tandis que les plus grandes sont adaptées aux circuits plus complexes.

Malgré leur polyvalence, les planches à pain ont aussi des limites. Elles ne sont pas adaptées aux très hautes fréquences ou aux circuits nécessitant une puissance élevée. En outre, les connexions sont parfois moins fiables que les connexions soudées, surtout si la planche à pain s'use avec le temps.

Dans l'ensemble, les planches à pain sont un outil essentiel pour toute personne travaillant dans le domaine de l'électronique, qu'il s'agisse de débutants apprenant les bases ou de développeurs expérimentés cherchant à créer des prototypes rapidement et efficacement. Elles sont l'équivalent électronique du carnet de croquis d'un artiste : un endroit où explorer des idées et expérimenter avant de créer l'œuvre finale.

5. PROJETS

Bienvenue dans le chapitre consacré aux projets électroniques innovants avec le Raspberry Pi Pico ! Dans cette section, vous découvrirez un large éventail d'applications allant du simple contrôle de LED au développement de systèmes plus complexes tels que des stations météorologiques automatisées et des systèmes d'éclairage dynamiques. Chaque projet est soigneusement conçu pour vous donner une expérience pratique avec une variété de composants matériels.

Commencez votre voyage de découverte avec des projets de base qui vous apprennent à utiliser les GPIO (General Purpose Input/Output) sur le Raspberry Pi Pico, et augmentez vos compétences avec des sujets plus avancés tels que le contrôle des servomoteurs ou l'utilisation de capteurs pour la surveillance de l'environnement. En utilisant des composants tels que des encodeurs rotatifs, des capteurs à ultrasons, des buzzers et des LED Neopixel, vous apprendrez à concevoir des systèmes interactifs et réactifs.

Chaque projet fournit une introduction détaillée aux composants requis, des instructions pas à pas sur la configuration du matériel et des exemples clairs de code de programme pour vous aider à comprendre les principes de l'électronique et de la programmation informatique. Il explique également comment intégrer des capteurs et des actionneurs externes pour collecter des données en temps réel et y répondre.

Ces projets ne sont pas seulement éducatifs, ils sont aussi amusants et offrent de nombreuses possibilités de personnalisation et d'extension afin que vous puissiez développer vos propres solutions créatives. Que vous soyez un débutant qui commence à explorer le monde de l'électronique numérique ou un développeur expérimenté qui cherche à étendre ses compétences, ce chapitre fournit les ressources et l'inspiration dont vous avez besoin pour améliorer vos compétences techniques et vous amuser en apprenant. Préparez-vous à développer vos compétences en programmation et en électronique en étant guidé à travers chaque projet tout en vous amusant à créer et à expérimenter.

Vous trouverez les codes d'exemple correspondants à la fin de chaque projet. Vous pouvez également télécharger les **fichiers ici**.

5.1 AFFICHAGE DE LA DISTANCE

Dans notre premier projet, notre objectif est de construire un télémètre à ultrasons qui visualise les distances sur notre écran TFT. Ce projet est une excellente introduction à la détection et à la visualisation de données avec votre Raspberry Pi Pico.

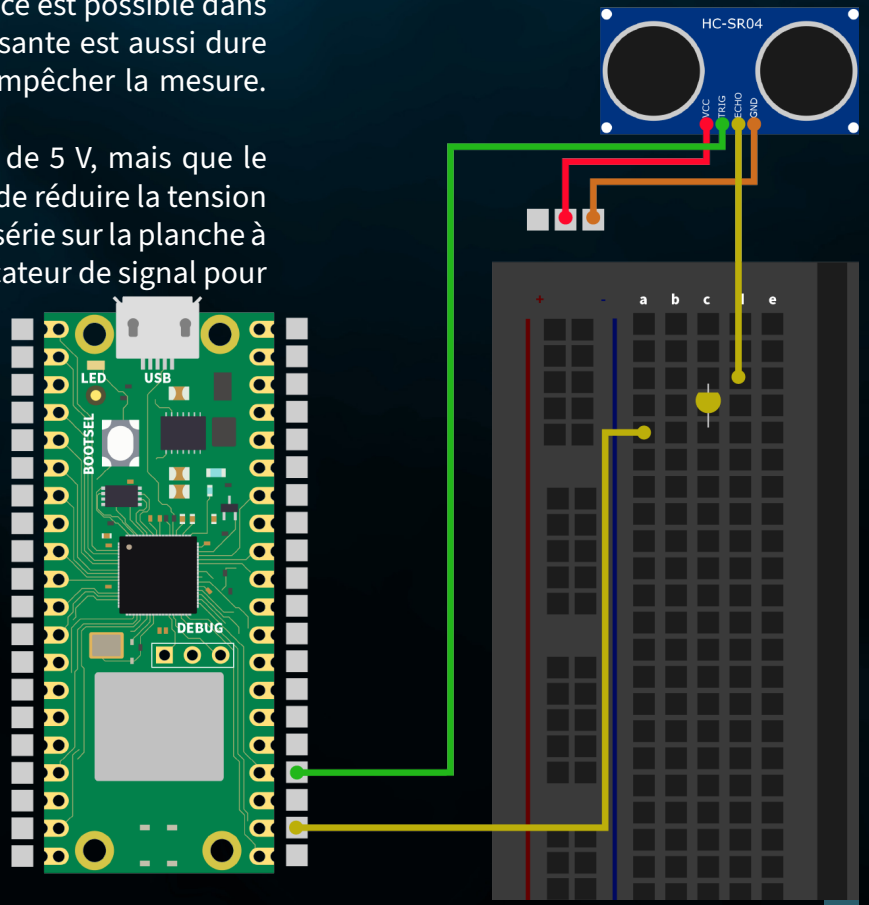
CAPTEUR À ULTRASONS: Un émetteur émet une onde ultrasonique et mesure le temps nécessaire pour qu'elle soit réfléchie et revienne à l'émetteur. La vitesse du son étant connue dans différents milieux tels que l'air (343 m/s à 20 °C) et l'eau (1 484 m/s), la distance jusqu'à la surface réfléchissante peut être calculée (en divisant par deux le temps de transit, puisque la distance aller-retour a été mesurée). Une impulsion du microcontrôleur sur l'entrée de déclenchement déclenche une séquence de huit courtes impulsions ultrasoniques. Dès que le signal est à nouveau reçu, la sortie écho passe brièvement à l'état haut. Le temps écoulé entre le déclenchement et le signal d'écho correspond au temps de transit. La mesure de la distance est possible dans une plage allant d'environ 2 cm à 400 cm et est très précise tant que la surface réfléchissante est aussi dure et régulière que possible. Les matériaux souples tels que la moquette peuvent même empêcher la mesure.

Étant donné que le capteur ultrasonique est un capteur qui nécessite une alimentation de 5 V, mais que le Raspberry Pi Pico ne peut traiter que des signaux de 3,3 V sans problème, il est nécessaire de réduire la tension du signal afin d'éviter tout dommage. Nous y parvenons en connectant notre LED jaune en série sur la planche à pain et en l'utilisant pour abaisser la tension de notre signal. La DEL sert également d'indicateur de signal pour le signal d'écho actif.

Pour plus de détails sur les DEL, voir le chapitre 5.5.

RASPBERRY PI PICO	CAPTEUR À ULTRASONS
3V3	VCC
GP17	Trig
GP16	Echo
GND	GND

ATTENTION! Pour ce projet, il est nécessaire de mettre les interrupteurs du relais et du DHT11 sur OFF et l'interrupteur de l'écran TFT sur ON.



RÉSUMÉ: Dans notre premier projet, nous mesurons des distances à l'aide du capteur à ultrasons et visualisons la distance mesurée en remplissant plus ou moins le graphique sur l'écran TFT. Dans notre exemple, nous remplissons complètement l'écran à partir d'une distance mesurée 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
```

Initialisation du capteur à ultrasons et de l'écran TFT

Fonction auxiliaire de réglage de la plage de valeurs

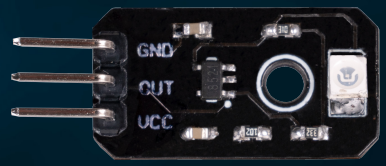
Mesure de la distance

```
# 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))
```

Calcul de la plage de valeurs et description de l'écran TFT

5.2 STATION MÉTÉO

Plongez dans le deuxième projet de notre aventure électronique et créez votre propre station météo ! Ce projet combine l'utilisation d'un capteur UV avec le capteur de température et d'humidité DHT11 pour non seulement vous donner un aperçu des conditions météorologiques actuelles, mais aussi mesurer le rayonnement UV à l'endroit où vous vous trouvez. Toutes ces informations sont clairement affichées sur l'écran TFT coloré, de sorte que vous pouvez voir la météo et le rayonnement UV d'un seul coup d'œil.



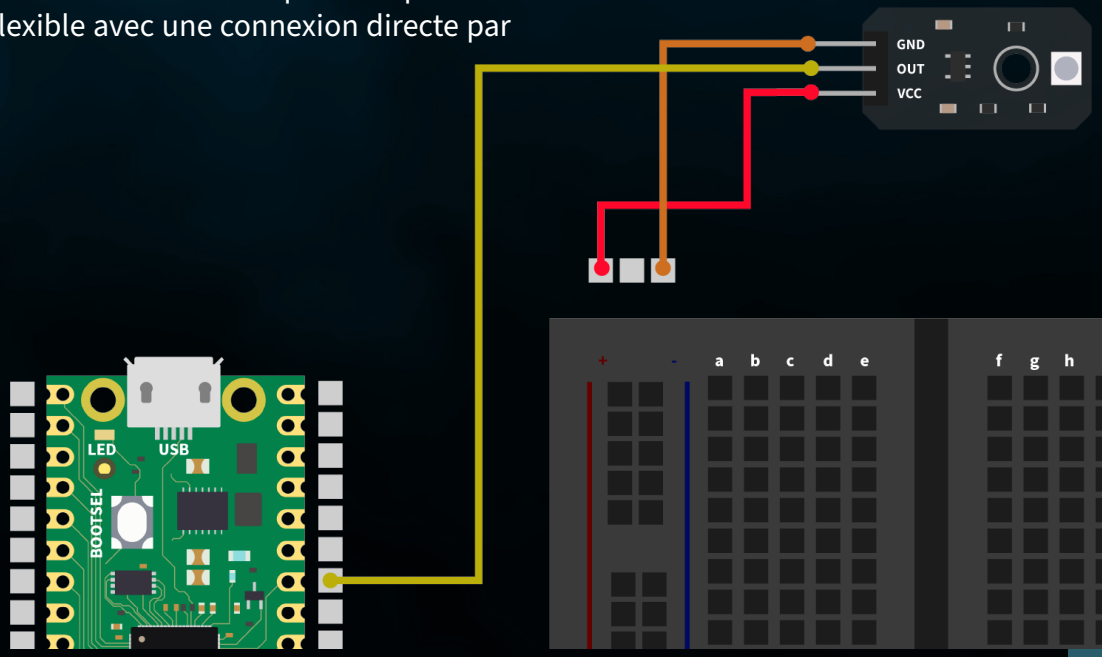
CAPTEUR UV: Le capteur UV est un petit composant qui nous aide à mesurer les rayons ultraviolets (UV) invisibles du soleil. Les rayons UV sont la partie invisible de la lumière du soleil, mais ils peuvent avoir un impact sur notre peau et notre santé. Pensez aux coups de soleil ou au bronzage de la peau - tous deux sont causés par les rayons UV.

Le capteur contient un matériau qui réagit aux rayons UV. Lorsque les rayons UV atteignent ce matériau, le capteur modifie sa résistance électrique. Ce changement est converti par le capteur en un signal que nous pouvons mesurer et lire. À l'aide du Raspberry Pi Pico, nous pouvons ensuite convertir ce signal en une valeur qui indique l'intensité du rayonnement UV actuel.

Connectez d'abord le capteur UV à votre Raspberry Pi Pico à l'aide des câbles fournis. Bien que vous puissiez également le brancher sur la planche à pain, vous êtes beaucoup plus flexible avec une connexion directe par câble.

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

ATTENTION! Pour ce projet, il est nécessaire de mettre l'interrupteur du relais sur OFF et les interrupteurs de l'écran TFT et du capteur DHT11 sur ON.



RÉSUMÉ : Notre station météorologique lit les capteurs DHT11 et UV et affiche les données sur l'écran 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)
```

Initialisation de l'écran TFT

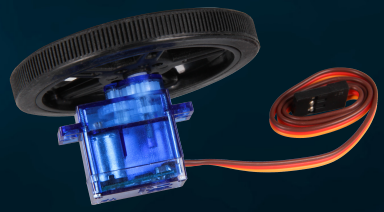
Mesure des valeurs des capteurs

Sortie sur l'écran

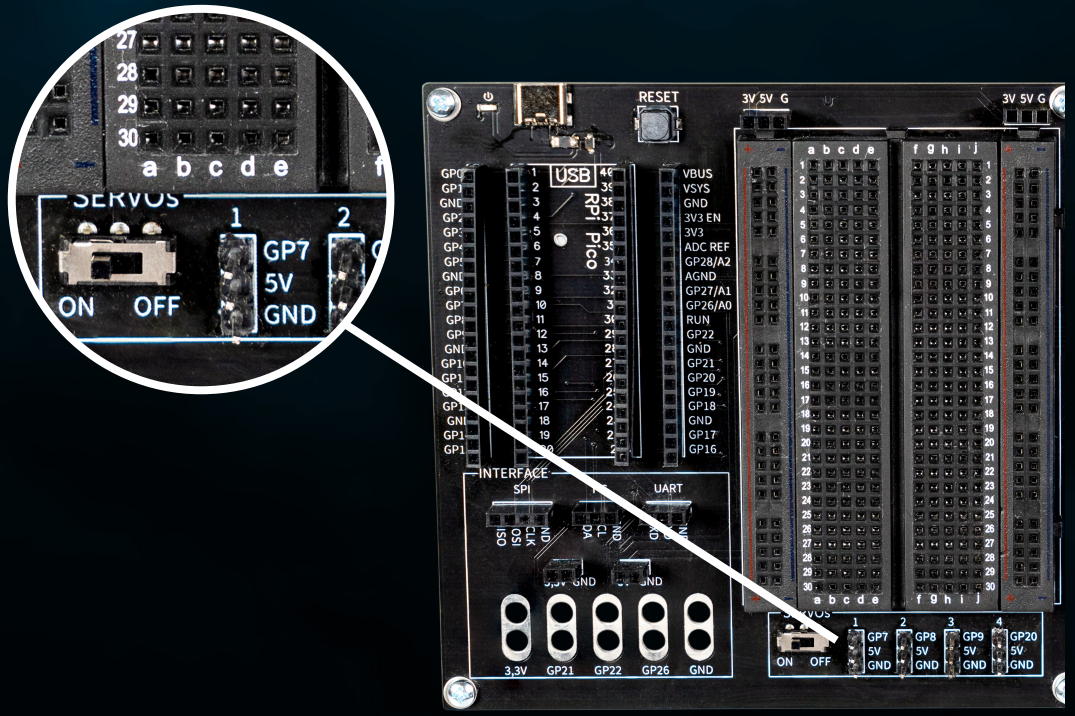
5.3 SERVOCOMMANDE

Bienvenue dans le troisième projet de notre série d'aventures électroniques passionnantes avec le kit d'exploration ! Cette fois-ci, il s'agit de mouvement et de contrôle. Notre objectif est de programmer et de contrôler un servomoteur de manière à ce que son sens de rotation puisse être contrôlé en appuyant simplement sur un bouton. Ce projet constitue non seulement une excellente introduction au monde de la commande de moteurs, mais il montre également comment renforcer les interactions grâce à un retour visuel sur un écran TFT.

SERVOMOTEUR : Un servomoteur se compose d'un moteur électrique, d'un réducteur et d'une électronique de commande. Du côté de la sortie du réducteur se trouve une roue dentée sur laquelle est montée la roue du servo. Les servos sont utilisés dans le modélisme, par exemple pour contrôler la position des ailes ou du gouvernail d'un avion ou d'un bateau. De plus en plus de servos sont également utilisés dans la construction automobile pour fermer automatiquement les portes, pour les lève-vitres, les rétroviseurs et d'autres éléments réglables.



Connectez d'abord le servomoteur à l'interface servo avec le numéro 1 sur votre Explorer Board.



ATTENTION! Pour ce projet, il est nécessaire de mettre l'interrupteur de l'écran TFT, des boutons et des servos sur ON.

RÉSUMÉ : Nous contrôlons notre servomoteur et le laissons passer de la rotation gauche à la rotation droite, contrôlées par nos boutons.

Initialisation du servomoteur et des boutons

```
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
```

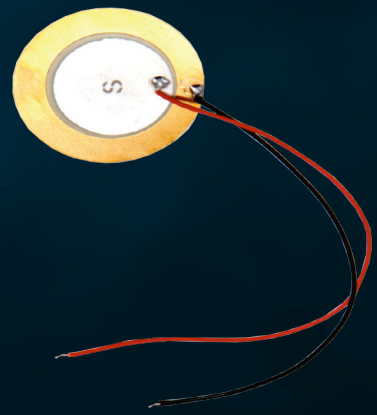
Vérification des boutons

```
    # 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
```

Servocommande

5.4 BUZZER FABRIQUÉ PAR NOS SOINS

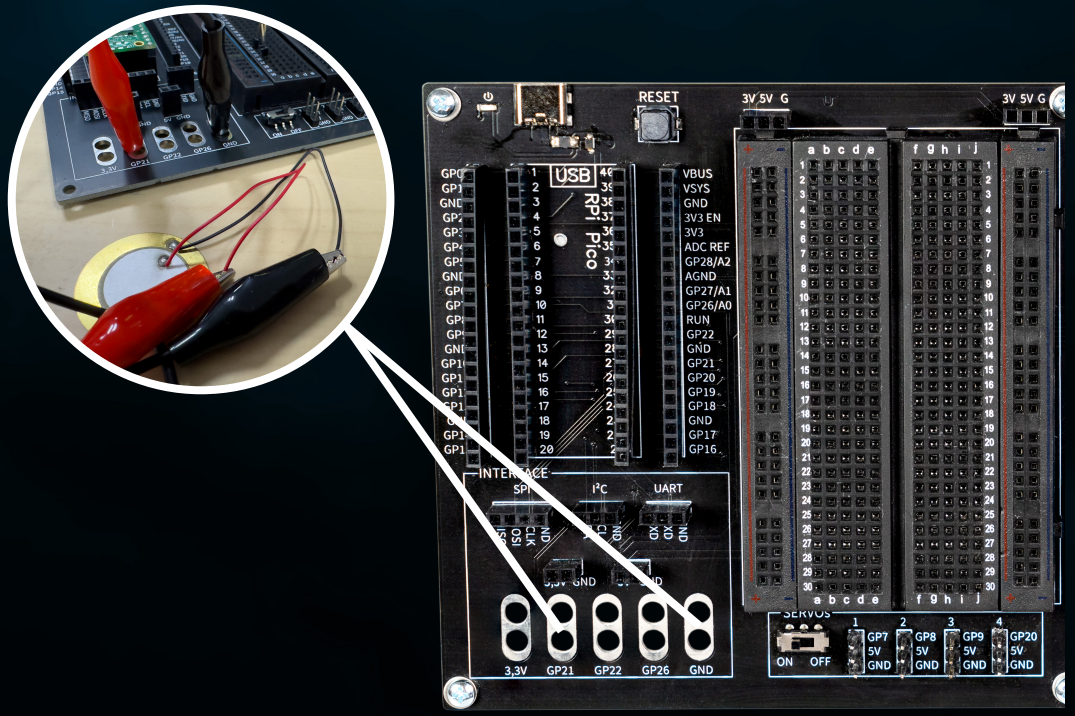
Le quatrième projet de notre aventure électronique avec le kit d'exploration porte sur le son ! Nous nous plongeons dans le monde des signaux acoustiques en créant notre propre circuit de sonnerie. Ce projet vous permet non seulement de comprendre les bases de la création de circuits, mais aussi d'apprendre à créer des signaux sonores à l'aide d'outils simples. L'utilisation de pinces crocodiles rend la construction particulièrement conviviale et accessible, même pour ceux qui n'en sont qu'au début de leur apprentissage de l'électronique.



Tout d'abord, nous connectons soigneusement le buzzer à la carte Explorer à l'aide de pinces crocodiles. Ces bornes sont idéales pour des connexions rapides et flexibles sans soudure. L'avertisseur, un petit composant capable de produire un son, devient la pièce maîtresse de notre projet. Lorsque le courant est appliqué, le buzzer vibre et produit un son.

Tout d'abord, connectez une pince crocodile au connecteur de pince crocodile GP21 de votre carte Explorer. Branchez l'autre extrémité de la pince crocodile sur le câble rouge de l'avertisseur. Connectez une autre pince crocodile à la connexion GND de la pince crocodile de votre carte Explorer. Connectez l'autre extrémité de la borne au câble noir de l'avertisseur.

RASPBERRY PI PICO	BUZZER
GP21	Câble rouge
GND	Câble noir



ATTENTION! Pour ce projet, il est nécessaire de mettre l'interrupteur des boutons sur ON.

RÉSUMÉ : Nous connectons un buzzer externe à notre Raspberry Pi Pico pour jouer une mélodie simple et amusante.

```
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
```

Liste des notes

Mémoire mélodique

Fonction pour jouer les notes

5.5 VOTRE PROPRE CIRCUIT

Dans le cinquième projet de notre aventure électronique avec le kit d'exploration, nous explorons le monde fascinant du contrôle de la lumière. Cette fois, nous construisons un circuit que vous pouvez utiliser pour contrôler des LED. C'est une occasion fantastique de comprendre les principes des circuits électroniques tout en contrôlant les effets de lumière.

LEDS: Les DEL, ou diodes électroluminescentes, sont des sources lumineuses petites mais puissantes qui sont utilisées dans de nombreux projets électroniques. Elles présentent de nombreux avantages par rapport aux ampoules traditionnelles, comme une durée de vie plus longue, une consommation d'énergie plus faible et la possibilité de s'allumer en différentes couleurs. Une DEL est constituée d'un matériau semi-conducteur qui émet de la lumière lorsqu'il est traversé par un courant électrique.

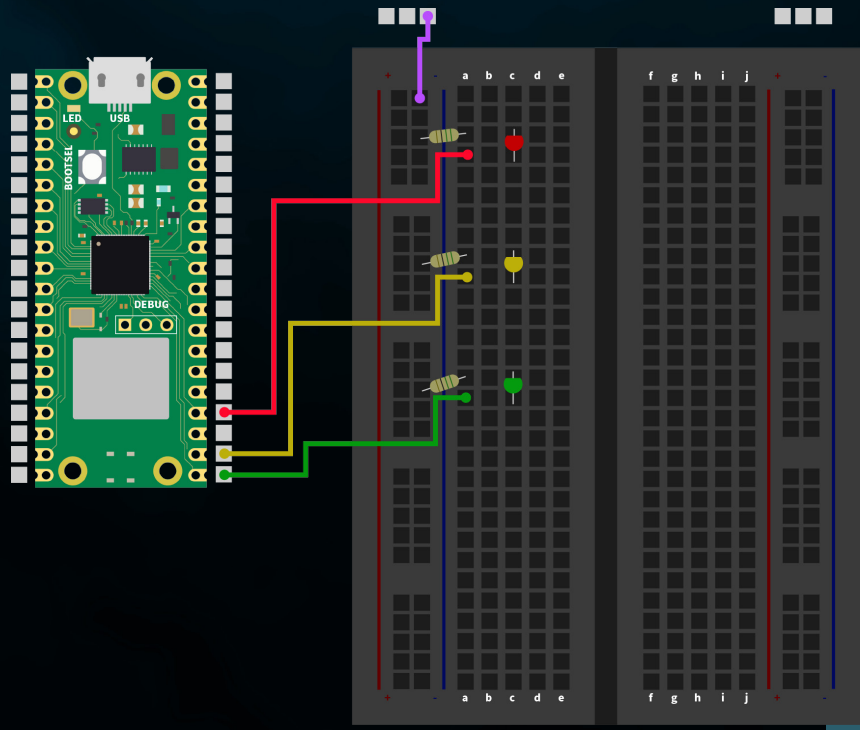
Il est important de respecter la polarité des DEL, car elles ne fonctionnent que si le courant les traverse dans le bon sens. Cela signifie que le pôle positif de la source d'alimentation doit être connecté à l'extrémité positive de la DEL et que le pôle négatif de la source d'alimentation doit être connecté à l'extrémité négative de la DEL.

C'est ainsi que l'on reconnaît la polarité d'une DEL :

Jambe la plus longue : pour la plupart des DEL, la jambe la plus longue est l'anode (+), c'est-à-dire la connexion positive. La branche la plus courte est la cathode (-), c'est-à-dire la connexion négative.

Bord plat : Il peut y avoir un bord plat sur le côté du boîtier de la DEL. Ce côté marque généralement la cathode, c'est-à-dire le pôle négatif.

Reconstituez d'abord le circuit comme indiqué dans le schéma suivant. Veillez toutefois à ce que la polarité des DEL soit correcte. Utilisez également une résistance en série de 56 Ω (vert-bleu-noir) pour chaque DEL.



RASPBERRY PI PICO	LEDS
GP18	LED rouge
GP17	LED jaune
GP16	LED verte

ATTENTION! For this project, it is necessary to set the switch for the **TFT DISPLAY** to **OFF**.

RÉSUMÉ: Nous contrôlons trois LED différentes via les broches du Raspberry Pi Pico, chaque LED clignotant alternativement.

Initialisation des DEL

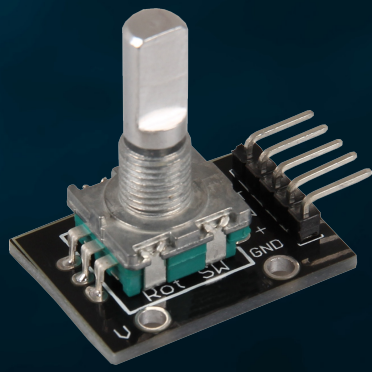
```
from machine import Pin
import utime

# 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
```

Fonction de clignotement de la LED

Boucle principale

5.6 CONTRÔLE DES LED

Dans le sixième projet de notre aventure électronique, nous utilisons notre encodeur rotatif pour contrôler la luminosité et la couleur des LED - une façon simple mais fascinante de s'immerger dans l'électronique. L'encodeur rotatif, notre élément de contrôle central, permet une interaction ludique grâce à sa double fonction. Les changements de luminosité sont contrôlés par la rotation, tandis qu'une pression sur l'encodeur fait défiler les couleurs des LED - idéal pour illustrer les bases de l'électronique et du mélange des couleurs.



CODEUR ROTATIF: Le codeur rotatif est un petit appareil astucieux qui convertit vos mouvements rotatifs en signaux électroniques. Imaginez un bouton rotatif comme celui que vous connaissez sur une radio. Lorsque vous tournez ce bouton, le codeur rotatif peut mesurer la distance et la direction dans lesquelles vous l'avez tourné. Cette information peut ensuite être utilisée, par exemple, pour modifier le volume, naviguer dans les menus ou, dans nos projets, régler la luminosité des LED.

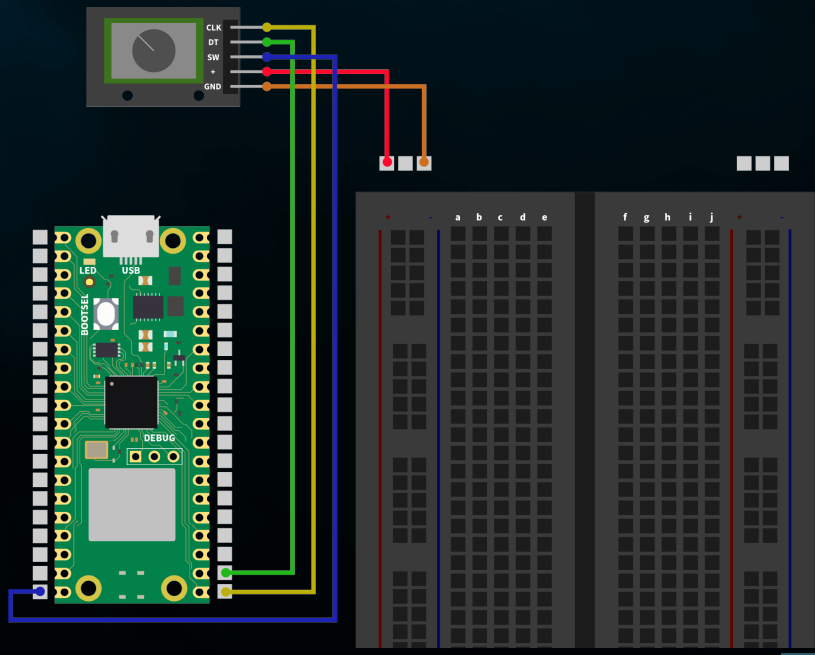
Les codeurs rotatifs sont souvent dotés d'un bouton intégré, ce qui signifie qu'ils peuvent également fonctionner comme un interrupteur à pression. Lorsque vous appuyez sur le bouton rotatif, le codeur rotatif reconnaît cette pression comme un signal distinct. Ce signal peut être utilisé pour diverses fonctions, telles que la mise en marche et l'arrêt d'un appareil ou le changement de mode de fonctionnement.

EN RÉSUMÉ: Un encodeur rotatif vous permet d'envoyer diverses commandes à vos projets électroniques en tournant et en appuyant. C'est un outil intuitif et polyvalent qui rend les interactions avec vos projets faciles et amusantes.

Connectez d'abord l'encodeur rotatif à votre Raspberry Pi Pico comme suit :

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

ATTENTION! Pour ce projet, il est nécessaire de mettre l'interrupteur de l'écran TFT sur OFF et l'interrupteur des LED sur ON.



RÉSUMÉ: Nous utilisons l'encodeur rotatif pour contrôler la couleur et la luminosité de nos quatre DEL. Tourner l'encodeur modifie la luminosité, tandis qu'appuyer sur l'encodeur ajuste la couleur des LED.

```
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()
```

Initialisation des LED et du codeur rotatif

Fonction du codeur rotatif

```
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 des notes

5.7 CONTRÔLE AUTOMATIQUE DE LA LUMINOSITÉ

Dans le septième projet de notre aventure électronique, nous utilisons une photodiode pour contrôler automatiquement la luminosité des DEL. La photodiode convertit la lumière en un signal électrique de sorte que les LED s'allument plus fort lorsqu'il fait sombre et s'éteignent lorsqu'il y a plus de lumière ambiante.

En connectant la photodiode à la carte Explorer et en la programmant sur le Raspberry Pi Pico, les LED s'adaptent intelligemment à la luminosité ambiante. Ce projet montre comment des systèmes électroniques réactifs et économes en énergie peuvent être construits avec des composants simples.

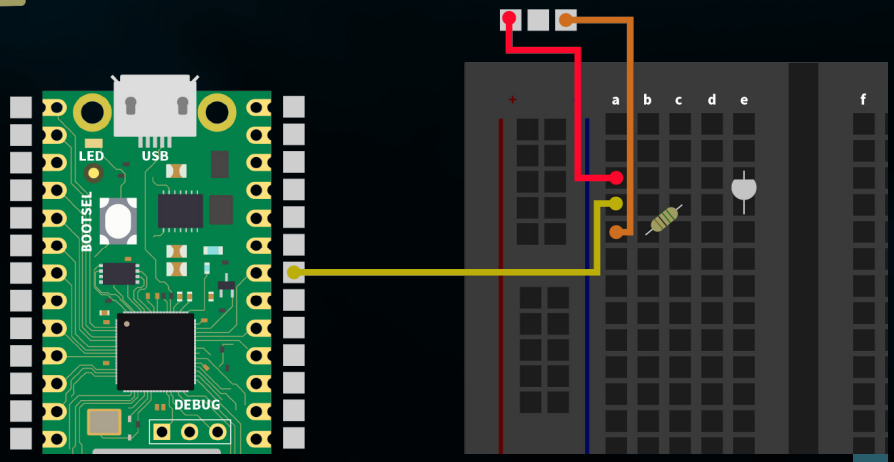
PHOTODIODE: Une photodiode est un type particulier de semi-conducteur qui réagit à la lumière qui l'atteint en générant un courant électrique. Imaginez une photodiode comme un petit panneau solaire : lorsque la lumière tombe dessus, elle convertit cette lumière en un signal électrique. Plus la lumière atteint la photodiode, plus le signal est fort.

Les photodiodes sont très sensibles et peuvent détecter même de petites quantités de lumière, ce qui les rend idéales pour les projets où il est important de mesurer la luminosité ou la présence de lumière. Elles peuvent par exemple être utilisées dans des régulateurs automatiques de luminosité, des capteurs de lumière ou dans le cadre d'un système de contrôle de l'éclairage.

En résumé, les photodiodes sont des détecteurs de lumière efficaces qui permettent de faire réagir intelligemment les appareils électroniques aux variations de l'éclairage ambiant.

Connectez d'abord la photodiode via la planche à pain comme suit. Notez que l'utilisation d'une résistance est également nécessaire ici. Utilisez ici la résistance de 100 kΩ (marron-noir-jaune).

RASPBERRY PI PICO	PHOTODIODE
3V3	Cathode positive
GP28/A2	Anode négative



ATTENTION! Pour ce projet, il est nécessaire de mettre l'interrupteur du relais sur OFF et l'interrupteur des LED sur ON.

RÉSUMÉ: Nous utilisons notre photodiode pour mesurer la luminosité ambiante et régler la luminosité de quatre DEL. L'intensité des LED change en fonction de la lumière détectée par la photodiode, les environnements plus sombres conduisant à des LED plus lumineuses et vice versa. Il est préférable d'utiliser une lampe de poche pour obtenir le meilleur résultat possible.

```
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)
```

Initialisation des LED et de la photodiode

Mesure de la photodiode et contrôle de la luminosité

5.8 CONTRÔLE DES LED RVB

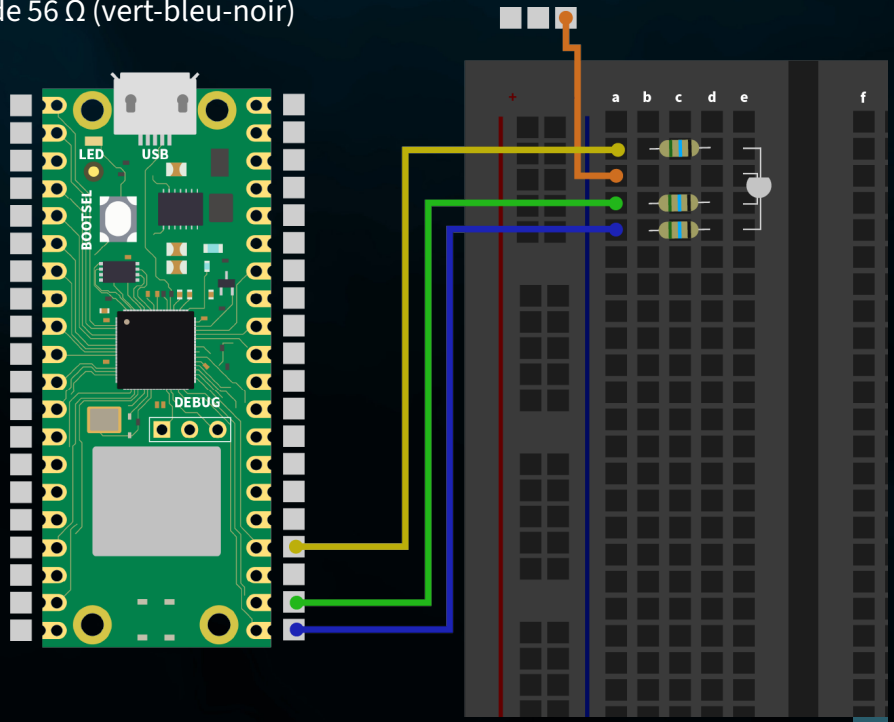
Dans le huitième et dernier projet de notre série sur l'électronique, nous allons nous concentrer sur le contrôle des couleurs des DEL RVB à l'aide des boutons intégrés à la carte Explorer. Les LED RVB sont des diodes électroluminescentes spéciales qui combinent la lumière rouge, verte et bleue (RVB) pour afficher une large gamme de couleurs. En réglant l'intensité de chaque composante de couleur individuellement, il est possible de créer presque n'importe quelle couleur.

Dans ce projet, nous connectons la LED RVB à la planche à pain et utilisons les boutons existants pour contrôler les couleurs de la LED. Chaque bouton est assigné à une couleur (rouge, vert, bleu).

RGB LED: Une LED RVB combine le rouge, le vert et le bleu en un seul point lumineux. En modifiant la luminosité de chacune des trois couleurs, il est possible de créer presque n'importe quelle couleur. Pour ce faire, la modulation de largeur d'impulsion (PWM) contrôle l'intensité de chaque couleur. Ainsi, les LED RVB à trois couleurs seulement permettent d'obtenir un large spectre de couleurs, idéal pour les projets d'éclairage colorés.

Connectez d'abord la DEL RVB à la plaque d'essai comme suit. Veuillez noter que chacun des trois canaux de couleur nécessite également une résistance en série ici. Vous devez utiliser la résistance de 56 Ω (vert-bleu-noir) ici.

RASPBERRY PI PICO	RGB-LED
GP18	Première épingle
GND	Deuxième épingle
GP17	Troisième goupille
GP16	Quatrième goupille



ATTENTION! Pour ce projet, il est nécessaire de mettre l'interrupteur du TFT sur OFF et celui des boutons sur ON.

RÉSUMÉ: Les trois canaux de couleur de la LED RVB (rouge, vert et bleu) sont activés et désactivés à l'aide des boutons (gauche, haut et droite).

```
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
```

Initialisation de la LED et des boutons

Test des boutons et contrôle de la LED

6. OBLIGATIONS D'INFORMATION ET DE REPRISE

NOS OBLIGATIONS D'INFORMATION ET DE REPRISE EN VERTU DE LA LOI ALLEMANDE SUR LES ÉQUIPEMENTS ÉLECTRIQUES ET ÉLECTRONIQUES (ELEKTROG)

SYMBOLE SUR LES ÉQUIPEMENTS ÉLECTRIQUES ET ÉLECTRONIQUES : 
Cette poubelle barrée signifie que les appareils électriques et électroniques ne doivent pas être jetés dans les ordures ménagères. Vous devez déposer les appareils usagés dans un point de collecte. Avant de les déposer, vous devez séparer les piles et les accumulateurs usagés qui ne sont pas contenus dans l'ancien appareil.

OPTIONS DE RETOUR :
En tant qu'utilisateur final, vous pouvez retourner votre ancien appareil (qui remplit essentiellement la même fonction que le nouvel appareil acheté chez nous) pour qu'il soit éliminé gratuitement lors de l'achat d'un nouvel appareil. Les petits appareils dont les dimensions extérieures ne dépassent pas 25 cm peuvent être éliminés avec les quantités normales de déchets ménagers, que vous ayez ou non acheté un nouvel appareil.

POSSIBILITÉ DE RETOUR DANS NOS LOCAUX PENDANT LES HEURES D'OUVERTURE :
SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

OPTION DE RETOUR DANS VOTRE RÉGION :
Nous vous enverrons un timbre pour colis avec lequel vous pourrez nous renvoyer l'appareil gratuitement. Pour ce faire, veuillez nous contacter par e-mail à l'adresse service@joy-it.net ou par téléphone.

INFORMATIONS SUR L'EMBALLAGE :
Veuillez emballer soigneusement votre ancien appareil pour le transport. Si vous n'avez pas de matériel d'emballage approprié ou si vous ne souhaitez pas utiliser le vôtre, veuillez nous contacter et nous vous enverrons un emballage adapté.

7. SOUTIEN

Nous sommes également à votre disposition après votre achat. Si des questions restent sans réponse ou si des problèmes surviennent, nous sommes également disponibles par courrier électronique, par téléphone et par le biais du système d'assistance par tickets.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Téléphone : +49 (0)2845 9360 – 50 (Lun. - jeu. : 09:00 - 17:00 ou heure, ven. : 09:00 - 14:30 ou heure)

Pour plus d'informations, veuillez consulter notre site web :

WWW.JOY-IT.NET