EXPLORER SET

RB-P-XPLR-SET

INHOUDSOPGAVE

1. ALGEMENE INFORMATIE

Beste klant, bedankt dat u voor ons product hebt gekozen. Hieronder laten we u zien waar u op moet letten tijdens de inbedrijfstelling en het gebruik.

Mocht u tijdens het gebruik onverhoopt problemen ondervinden, aarzel dan niet om contact met ons op te nemen.

2. OVERZICHT APPARAAT & PINTOEWIJZING

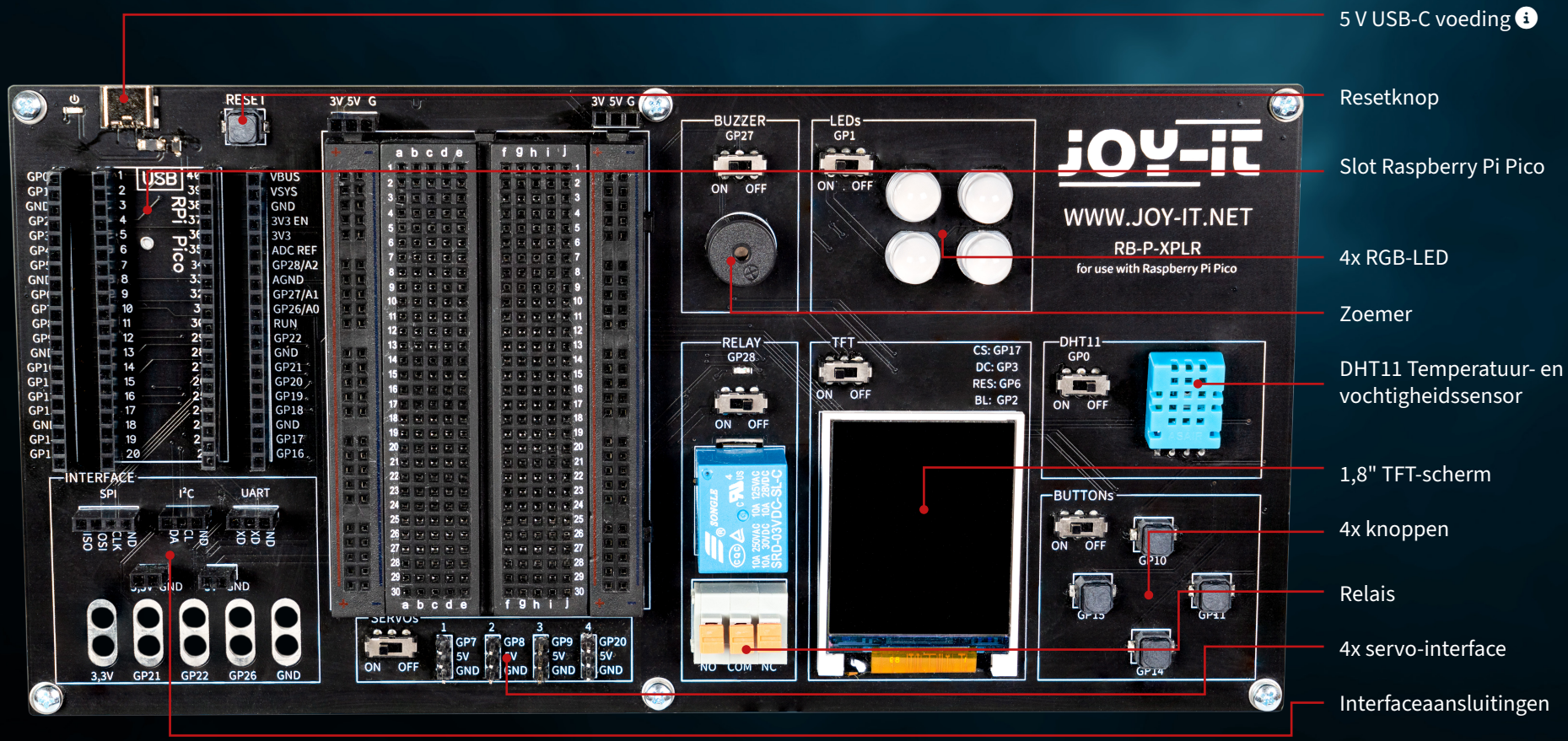
Ons Explorer Board is de eenvoudige en efficiënte manier om je Raspberry Pi Pico projecten te ontwikkelen.

Met de belangrijkste componenten al geïntegreerd, bespaar je tijd en moeite bij het bedraden. Het Explorer Board heeft een breed scala aan interface-aansluitingen, zodat u uw projecten kunt aansluiten op een verscheidenheid aan modules en apparaten. Met het geïntegreerde breadboard kunt u snel uw eigen projecten bouwen en realiseren.

Dankzij de mogelijkheid om alle modules afzonderlijk aan of uit te zetten, kunt u uw pinnen, die ook afzonderlijk naar buiten zijn geleid, op elk gewenst moment voor andere projecten gebruiken of op het geïntegreerde breadboard experimenteren.

Alle ingebouwde componenten kunnen via de betreffende schakelaar worden uitgeschakeld als ze niet nodig zijn. Dit betekent dat de bijbehorende pinnen indien nodig ook voor andere componenten kunnen worden gebruikt.

Links en rechts van de Raspberry Pi Pico zijn alle pinnen extra vormgegeven. Componenten kunnen hier direct worden aangesloten of via extra kabels naar het geïntegreerde breadboard worden geleid.

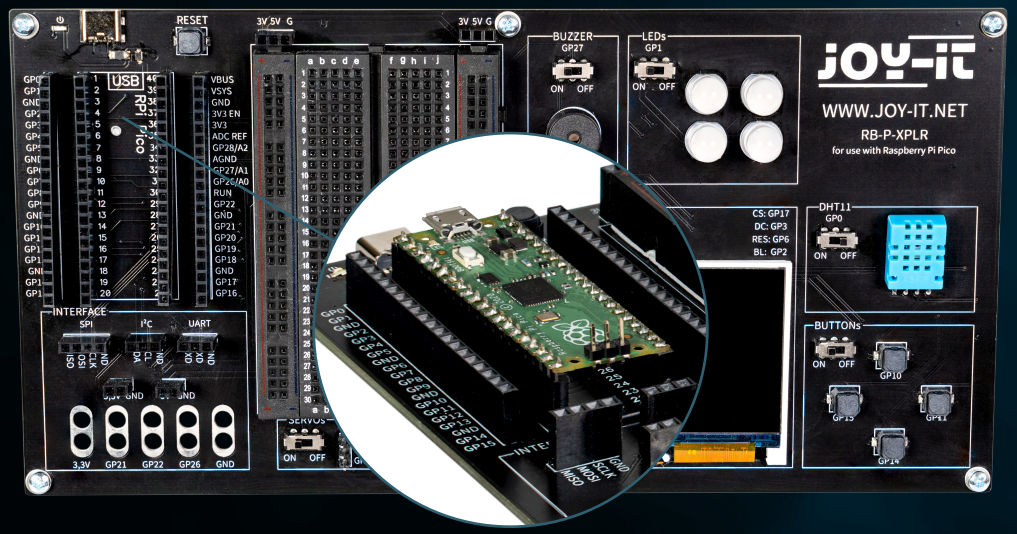


Houd er rekening mee dat de USB-C aansluiting altijd aangesloten moet zijn voor gebruik. Voeding via de micro-USB-aansluiting van de Raspberry Pi Pico is niet mogelijk.

PINBEZETTING	
Zoemer	GP27
LEDs	GP1
Relais	GP28
1,8" TFT-scherm	CS: GP17, DC: GP3, RES: GP6, BL: GP2
DHT11	GP0
Knoppen	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

Steek eerst je Raspberry Pi Pico in de sleuf op je bord.



Sluit nu een micro USB-kabel aan op je computer en op de Raspberry Pi Pico om te programmeren.

ATTENTIE! De USB-C poort op het Explorer bord wordt uitsluitend gebruikt voor de stroomvoorziening. Hij wordt niet gebruikt om gegevens over te brengen naar de Raspberry Pi. Je kunt een geschikt ontwikkelprogramma naar keuze gebruiken om ons voorbeeldprogramma over te zetten. Wij raden de **Thonny Python IDE aan**.

ATTENTIE! Als je nieuw bent in de wereld van microcontrollers en elektronica, maak je dan geen zorgen! We hebben een speciale beginnershandleiding voor je gemaakt. Deze gids is speciaal afgestemd op de behoeften van beginners en legt stap voor stap uit hoe je de Raspberry Pi Pico gebruikt.

Van basisconfiguratie tot het uitvoeren van projecten, in deze handleiding begeleiden we je door het hele proces. Onze gids bevat eenvoudig te begrijpen uitleg en nuttige tips om je te helpen snel en effectief je vaardigheden op schaal te ontwikkelen met de Raspberry Pi Pico. Je kunt onze gids **hier downloaden**.

4. MODULES IN DETAIL

Hieronder worden alle modules die beschikbaar zijn op het Explorer Board afzonderlijk uitgelegd met voorbeeldcodes. Hier kun je alle voorbeeldcodes en bibliotheken downloaden, evenals een voorbeeldcode die alle modules aan elkaar koppelt.

Voor het gebruik van sommige modules worden externe bibliotheken en een lettertypebestand gebruikt. Download de bibliotheken en laad ze in de lib-map van je Raspberry Pi Pico. Plaats het lettertypebestand in de hoofdmap van je Raspberry Pi Pico.

4.1 ZOEMER

Een buzzer produceert een signaaltoon, vergelijkbaar met een luidspreker. In tegenstelling tot een luidspreker is hij echter alleen geschikt voor een beperkt frequentiebereik, waardoor hij geen goed geluid produceert voor het weergeven van muziek of spraak. Hij is echter ideaal voor het genereren van luide waarschuwingstonen in de vorm van pieptonen. Wanneer een elektrisch apparaat een waarschuwingstoon produceert, is dat bijna altijd een zoemer. Bijvoorbeeld in wekkers, rookmelders of de veiligheidsgordelverklikker in auto's.

De zoemer is verbonden met GPIO pin 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-LED'S

RGB LED's zijn een type lichtemitterende diode die rood, groen en blauw combineren om een verscheidenheid aan kleuren te produceren. Net zoals een zoemer alleen eenvoudige tonen produceert, kunnen RGB LED's geen complexe beelden weergeven, maar ze zijn uitstekend in het mengen en variëren van kleuren. Elke LED in een RGB-eenheid kan worden gevarieerd in intensiteit om verschillende tinten te produceren, van zachte pasteltinten tot heldere, verzadigde kleuren. Dit maakt ze ideaal voor sfeerverlichting, decoratieve verlichting en in toepassingen waar visuele signalen nodig zijn, zoals in gamingopstellingen of als statusindicatoren in elektronische apparaten. Door hun veelzijdigheid en energiezuinigheid zijn ze een populaire keuze geworden in moderne verlichtingssystemen, hoewel ze door hun eenvoudige werking, net als de buzzer, geen complexe beelden of patronen kunnen maken zonder extra besturingseenheden.

De GPIO-LED's zijn verbonden met de GPIO-pin 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

Relais behoren tot de oudste elektromechanische componenten en werken als elektrisch gestuurde schakelaars. Met een kleine ingangsspanning en een lage stroom kan een grote elektrische belasting worden in- en uitgeschakeld aan de uitgang. Als het relais doorschakelt, gaat ook de rode LED branden. Je kunt gestripte kabeleinden in de aansluitbus steken (door de oranje hendel omlaag te drukken) om de drie aansluitingen te gebruiken.

Het relais is verbonden met GPIO-pin 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

Het LCD TFT-scherm met ongeveer 65.000 kleuren en een diagonaal van 1,8 inch heeft een resolutie van 128×160 pixels en kan worden bestuurd via SPI. Het is geschikt voor het weergeven van kleurrijke afbeeldingen. Letters en andere tekens worden weergegeven als afbeeldingen die bestaan uit vele afzonderlijke puntjes.

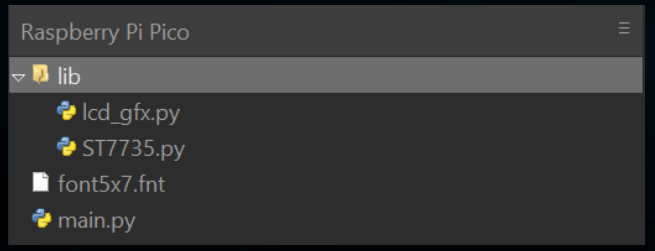
De TFT is aangesloten op de GPIO-pinnen GP17 (CS), GP3 (DC), GP6 (RES) en GP2 (BL).

```
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!‘)
```

Naast teksten kunnen bijvoorbeeld ook rechthoeken worden weergegeven:

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

ATTENTIE! Twee aparte bibliotheekbestanden en een lettertypebestand zijn nodig voor het TFT-scherm; je kunt de benodigde bestanden hier downloaden. Zet vervolgens alle bestanden uit de map Libraries over naar de hoofdmap van je Raspberry Pi Pico zodat de mapstructuur er als volgt uitziet:



4.5 DHT 11

De DHT11-sensor kan temperaturen van 0 °C tot 50 °C (nauwkeurigheid ±2 °C) en relatieve vochtigheid van 20 % tot 80 % (±5 %) detecteren (maximaal eenmaal per seconde). Weerstations zijn waarschijnlijk het primaire toepassingsgebied voor een sensor als de DHT11. Om de functionaliteit te testen, is het voldoende om je mond dicht bij de sensor te houden en langzaam uit te ademen. De ademlucht verschilt qua temperatuur en vochtigheid van de omgeving, wat zou moeten leiden tot een significante verandering in de waarden.

De DHT11 is verbonden met de GPIO-pin 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 BUTTONS

Knoppen zijn interactieve elementen in gebruikersinterfaces die een eenvoudige maar essentiële functie vervullen: gebruikersinvoer. Net zoals RGB-LED's een verscheidenheid aan kleuren kunnen weergeven, worden knoppen gebruikt om een breed scala aan commando's en acties in digitale omgevingen te starten.

De knoppen zijn verbonden met de GPIO-pinnen GP10 (boven), GP11 (rechts), GP14 (onder) en GP15 (links).

```
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

Een servo bestaat uit een elektromotor met tandwielkast en besturingselektronica. Aan de uitgaande kant van de tandwielkast zit een tandwiel waarop de servohoorn is gemonteerd. Servo's worden gebruikt in de modelbouw, bijvoorbeeld om de vleugel- of roerstand van een vliegtuig of schip te regelen. Ook in de autotechniek worden steeds meer servo's gebruikt voor het automatisch sluiten van deuren, voor raamregelaars, spiegels en andere verstelbare elementen.

De servo-aansluitingen zijn de GPIO-pinnen GP7, GP8, GP9 en 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

Interfaceverbindingen spelen een cruciale rol in de wereld van elektronica, vergelijkbaar met knoppen in gebruikersinterfaces. Ze maken communicatie en voeding tussen verschillende elektronische componenten mogelijk. De volgende aansluitingen zijn daarom te vinden in de interfacezone op ons Explorer Board:

SPI (Serial Perhipheral Interface): Deze verbinding wordt gebruikt voor snelle seriële gegevensoverdracht. Ze bestaat typisch uit vier lijnen: MISO (Master In, Slave Out), MOSI (Master Out, Slave In), SCK (Serial Clock) en SS (Slave Select). SPI is ideaal voor situaties waarin een hoge gegevensoverdrachtsnelheid vereist is, zoals bij het aansturen van LCD-schermen of SD-kaarten.

I2C (Inter-Integrated Circuit): I2C is een tweedraads interface bestaande uit een datalijn (SDA) en een kloklijn (SCL). Het wordt vaak gebruikt in microcontrollertoepassingen voor communicatie tussen verschillende geïntegreerde schakelingen. Door zijn eenvoud is het ideaal voor toepassingen waar niet veel GPIO-pinnen beschikbaar zijn.

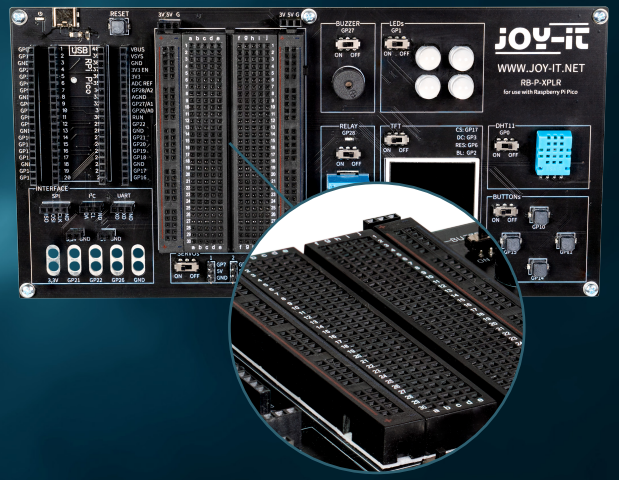
UART (Universal Asynchronous Receiver/Transmitter): Deze interface maakt asynchrone seriële communicatie mogelijk via twee lijnen: TX (zenden) en RX (ontvangen). UART wordt vaak gebruikt voor communicatie tussen microcontrollers en computers of voor het aansluiten van modules zoals GPS-ontvangers of Bluetooth-modules.

3,3 V en 5 V aansluitingen: Deze aansluitingen leveren de voeding voor elektronische componenten. 3,3 V wordt vaak gebruikt voor moderne microcontrollers en sensoren, terwijl 5 V vaak wordt aangetroffen in oudere of meer energievretende apparaten.

Aansluitingen voor krokodillenklemmen: Deze connectoren zijn ideaal voor tijdelijke verbindingen of voor testdoeleinden. Ze maken een snelle en eenvoudige aansluiting op verschillende componenten of meetapparaten mogelijk zonder te solderen. Er zijn in totaal vijf van deze connectoren op het Explorer Board, die flexibel kunnen worden gebruikt voor verschillende toepassingen.

Elk van deze aansluitingen heeft zijn specifieke toepassing en betekenis in de elektronica, net zoals verschillende soorten knoppen in een gebruikersinterface verschillende functies hebben. Ze bieden de nodige flexibiliteit en functionaliteit voor het opzetten en uitbreiden van elektronische systemen.

4.9 BROODPLANK



Breadboards zijn een onmisbaar hulpmiddel in de wereld van elektronica, net zoals interfaceconnectoren cruciaal zijn voor het verbinden van verschillende componenten. Hiermee kunnen elektronische schakelingen snel en zonder solderen worden gebouwd en getest, waardoor ze vooral populair zijn voor prototyping en educatieve doeleinden.

Een breadboard bestaat meestal uit een rechthoekig plastic blok met een groot aantal gaten die in rijen zijn aangebracht. Deze gaten zijn intern verbonden door metalen sporen die het mogelijk maken om componenten en draden eenvoudig aan te sluiten. De standaard layout van een breadboard bevat twee hoofdgebieden:

De belangrijkste gebieden: Deze bestaan uit een reeks parallelle rijen gaten, meestal gescheiden door een centrale groef. De gaatjes binnen een rij zijn elektrisch met elkaar verbonden. Deze opstelling is ideaal voor het inbrengen van geïntegreerde circuits (IC's) en andere componenten.

De stekkerdozen: Aan de rand van het breadboard zitten meestal een of twee rijen gaten die dienen als stroomstrips. Deze zijn verticaal verbonden over de hele lengte van het breadboard en bieden een handige manier om stroom en aarde te voorzien op verschillende punten op het circuit.

De flexibiliteit van een breadboard ligt in de herbruikbaarheid en de mogelijkheid om schakelingen te bouwen zonder permanente wijzigingen. Dit maakt het ideaal voor experimenten omdat fouten gemakkelijk kunnen worden gecorrigeerd en componenten kunnen worden verwijderd zonder schade. Het is ook een uitstekend leermiddel omdat het begrip van circuitlogica en componentfuncties op een praktische, visuele manier bevordert.

Bovendien zijn breadboards verkrijgbaar in verschillende maten en met verschillende aantallen aansluitpunten om aan verschillende eisen te voldoen. Kleinere breadboards zijn goed voor eenvoudige projecten en experimenten, terwijl grotere geschikt zijn voor complexere schakelingen.

Ondanks hun veelzijdigheid hebben breadboards ook beperkingen. Ze zijn niet geschikt voor zeer hoge frequenties of voor schakelingen die veel vermogen vereisen. Ook zijn de verbindingen soms minder betrouwbaar dan gesoldeerde verbindingen, vooral als het breadboard na verloop van tijd verslijt.

In het algemeen zijn breadboards een essentieel gereedschap voor iedereen die met elektronica werkt - van beginners die de basis leren tot ervaren ontwikkelaars die snel en efficiënt prototypes willen maken. Ze zijn het elektronische equivalent van het schetsboek van een kunstenaar: een plek om ideeën te verkennen en te experimenteren voordat het uiteindelijke werk wordt gemaakt.

5. PROJECTEN

Welkom bij het hoofdstuk over innovatieve elektronicaprojecten met de Raspberry Pi Pico! In dit hoofdstuk maak je kennis met een breed scala aan toepassingen, variërend van het eenvoudig aansturen van LED's tot het ontwikkelen van complexere systemen zoals geautomatiseerde weerstations en dynamische verlichtingssystemen. Elk project is zorgvuldig ontworpen om je hands-on ervaring te geven met een verscheidenheid aan hardwarecomponenten.

Begin je ontdekkingsreis met basisprojecten die je leren hoe je GPIO's (General Purpose Input/Output) op de Raspberry Pi Pico gebruikt, en vergroot je vaardigheden met meer geavanceerde onderwerpen zoals het aansturen van servomotoren of het gebruik van sensoren voor omgevingsbewaking. Met behulp van componenten zoals roterende encoders, ultrasone sensoren, zoemers en Neopixel LED's leer je interactieve en reactieve systemen te ontwerpen.

Elk project biedt een gedetailleerde introductie van de benodigde componenten, stapsgewijze instructies voor de hardwareconfiguratie en duidelijke voorbeelden van programmacode om je te helpen de principes van elektronica en computerprogrammering te begrijpen. Er wordt ook uitgelegd hoe je externe sensoren en actuatoren kunt integreren om real-time gegevens te verzamelen en erop te reageren.

Deze projecten zijn niet alleen leerzaam, maar ook leuk, met veel mogelijkheden voor aanpassingen en uitbreidingen zodat je je eigen creatieve oplossingen kunt ontwikkelen. Of je nu een beginner bent die net begint met het verkennen van de wereld van digitale elektronica of een ervaren ontwikkelaar die zijn vaardigheden wil uitbreiden, dit hoofdstuk biedt de bronnen en inspiratie die je nodig hebt om je technische vaardigheden te verbeteren en plezier te beleven aan het leren. Bereid je voor op het uitbreiden van je programmeer- en elektronicavaardigheden terwijl je door elk project wordt geleid terwijl je plezier beleeft aan het creëren en experimenteren.

Aan het einde van elk project vind je de bijbehorende voorbeeldcodes. Je kunt de bestanden **hier ook downloaden**.

5.1 AFSTANDSWEERGAVE

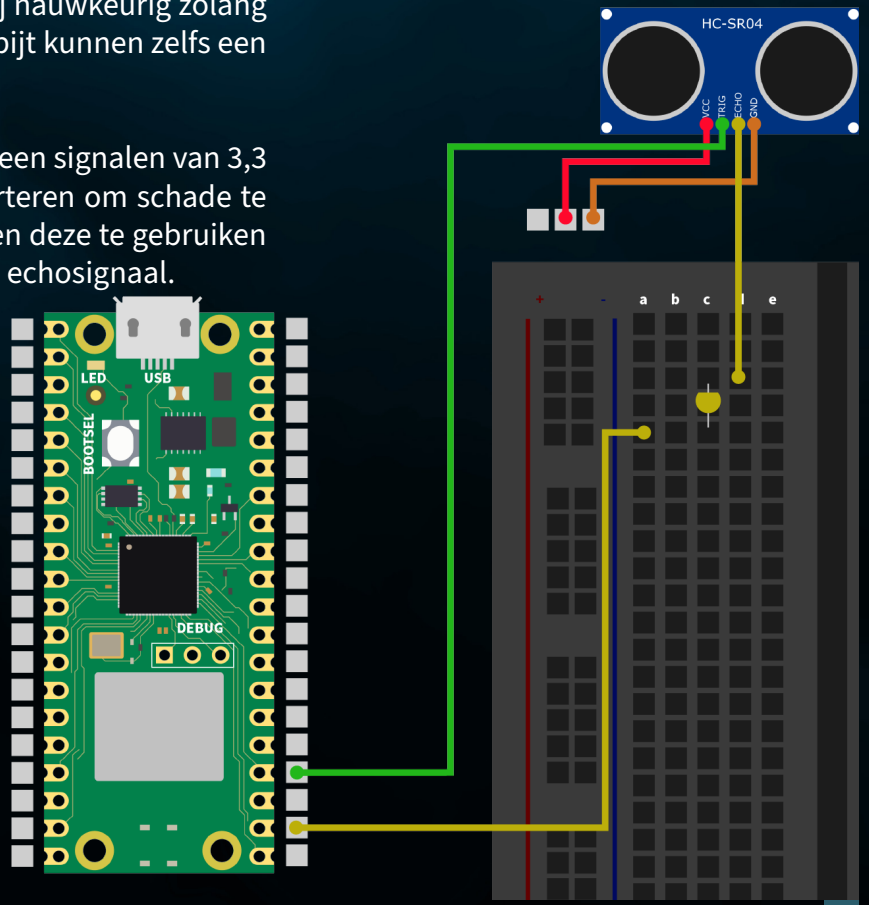
In ons eerste project willen we een ultrasone afstandsmeter bouwen die afstanden visualiseert op ons TFT-scherm. Dit project is een goede introductie in het detecteren en visualiseren van gegevens met je Raspberry Pi Pico.

ULTRASONE SENSOR: Een zender zendt een ultrasone golf uit en meet de tijd totdat deze wordt gereflecteerd en weer bij de zender aankomt. Aangezien de geluidssnelheid bekend is in verschillende media zoals lucht (343 m/s bij 20 °C) en water (1.484 m/s), kan de afstand tot het reflecterende oppervlak worden berekend (halvering van de transittijd, aangezien de afstand heen en terug is gemeten). Een puls van de microcontroller op de triggeringang activeert een reeks van acht korte ultrasone pulsen. Zodra het signaal weer wordt ontvangen, gaat de echo-uitgang kortstondig hoog. De tijd tussen de trigger en het echosignaal komt overeen met de transittijd. Afstandsmeting is mogelijk in het bereik van ongeveer 2 cm tot 400 cm en is vrij nauwkeurig zolang het reflecterende oppervlak zo hard en gelijkmatig mogelijk is. Zachte materialen zoals tapijt kunnen zelfs een meting verhinderen.

Omdat de ultrasone sensor een voeding van 5 V nodig heeft, maar de Raspberry Pi Pico alleen signalen van 3,3 V zonder problemen kan verwerken, is het nodig om de signaalspanning te down-converteren om schade te voorkomen. We bereiken dit door onze gele LED in serie te schakelen op het breadboard en deze te gebruiken om ons signaal om te zetten. De LED functioneert ook als signaalindicator voor het actieve echosignaal.

Meer informatie over LED's is te vinden in hoofdstuk 5.5.

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



ATTENTIE! Voor dit project is het noodzakelijk om de schakelaars voor het relais en de DHT11 op OFF te zetten en de schakelaar voor het TFT-scherm op ON.

SAMENVATTING: In ons eerste project meten we afstanden met de ultrasone sensor en visualiseren we de gemeten afstand door de grafiek op het TFT-scherm in meer of mindere mate te vullen. In ons voorbeeld vullen we het scherm volledig vanaf een gemeten afstand van 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
```

Initialisatie van de ultrasone sensor en het TFT-scherm

Hulpfunctie voor het aanpassen van het waardebereik

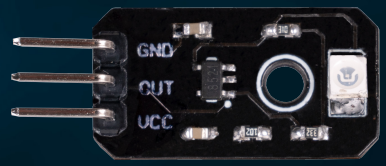
Afstandsmeting

```
# 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))
```

Berekening van het waardebereik
en beschrijving van het TFT-scherm

5.2 WEERSTATION

Duik in het tweede project van ons elektronica-avontuur waarbij je je eigen weerstation maakt! Dit project combineert het gebruik van een UV-sensor met de DHT11 temperatuur- en vochtigheidssensor om je niet alleen inzicht te geven in de huidige weersomstandigheden, maar ook de UV-straling op jouw locatie te meten. Al deze informatie wordt duidelijk weergegeven op het kleurrijke TFT-scherm, zodat je het weer en de UV-straling in één oogopslag kunt zien.



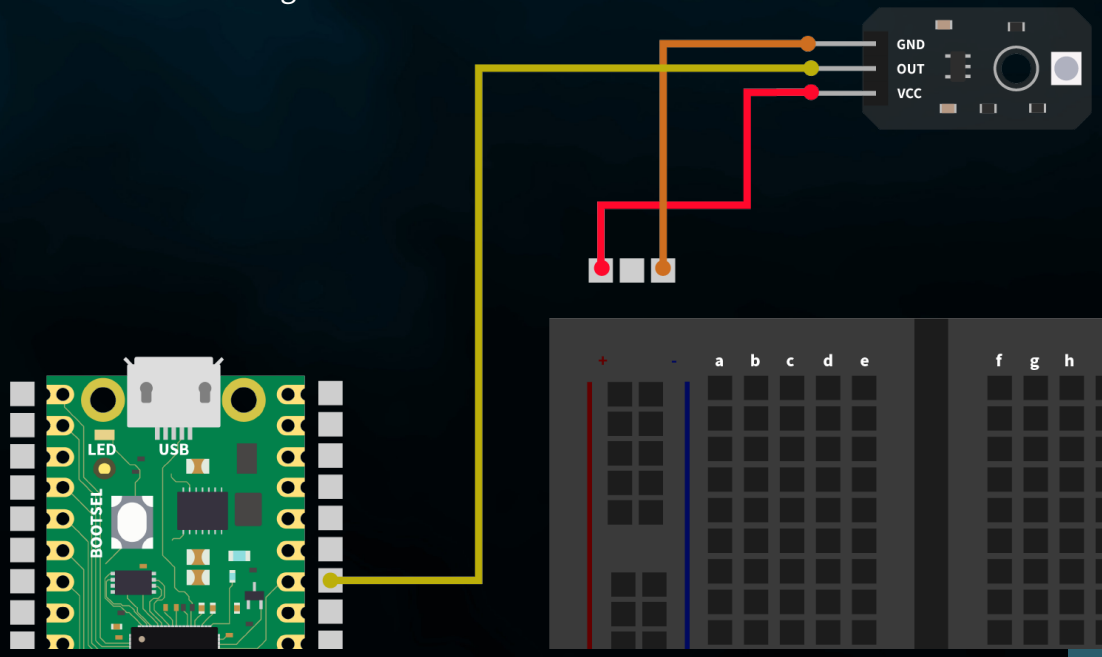
UV-SENSOR: De UV-sensor is een klein onderdeel dat ons helpt om de onzichtbare ultraviolette (UV) stralen van de zon te meten. UV-stralen zijn het deel van zonlicht dat onzichtbaar is, maar een impact kan hebben op onze huid en gezondheid. Denk aan zonnebrand of bruining van de huid - beide worden veroorzaakt door UV-stralen.

In de kern bevat de sensor een materiaal dat reageert op UV-straling. Wanneer UV-stralen dit materiaal raken, verandert de elektrische weerstand van de sensor. Deze verandering wordt door de sensor omgezet in een signaal dat we kunnen meten en uitlezen. Met behulp van de Raspberry Pi Pico kunnen we dit signaal omzetten in een waarde die aangeeft hoe sterk de UV-straling op dat moment is.

Sluit eerst de UV-sensor aan op je Raspberry Pi Pico met de bijgeleverde kabels. Hoewel je hem natuurlijk ook op het breadboard kunt aansluiten, ben je hier veel flexibeler met een directe kabelverbinding.

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

ATTENTIE! Voor dit project is het nodig om de schakelaar voor het relais op OFF te zetten en de schakelaars voor het TFT-scherm en de DHT11-sensor op ON.



SAMENVATTING: Ons weerstation leest de DHT11 en UV-sensoren en geeft de gegevens weer op het TFT-display.

```
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)
```

Initialisatie van het TFT-scherm

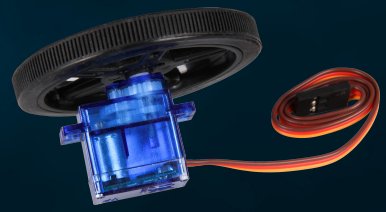
De sensorwaarden meten

Uitgang op het display

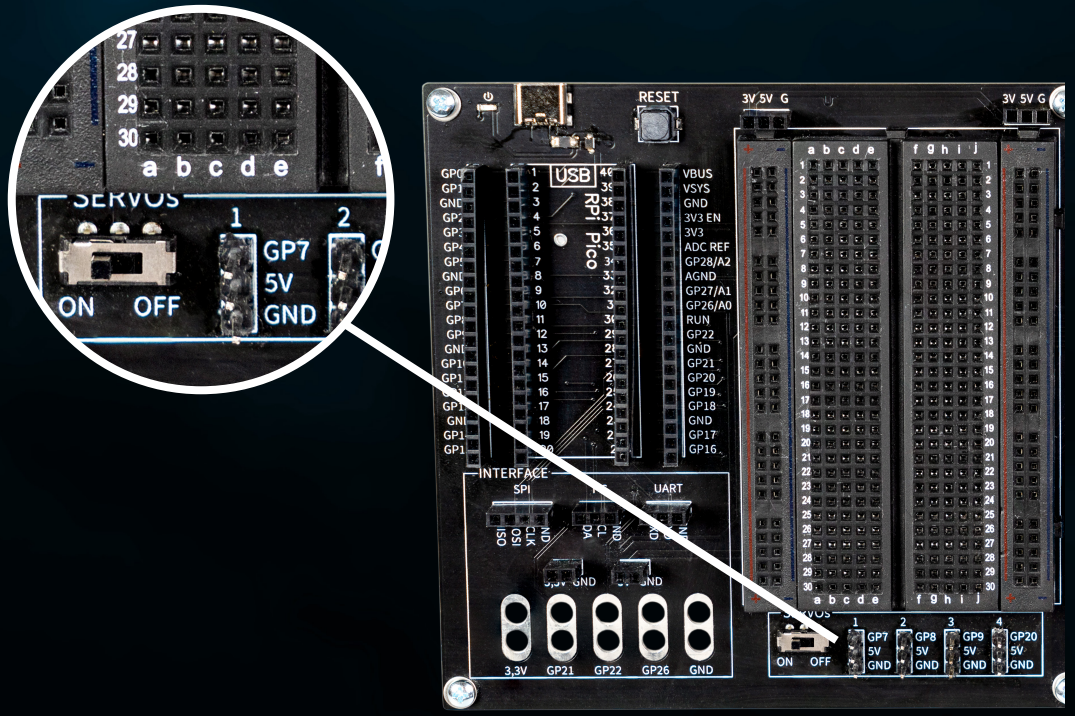
5.3 SERVOBESTURING

Welkom bij het derde project in onze reeks spannende elektronica-avonturen met de Explorer Set! Deze keer draait alles om beweging en besturing. Ons doel is het programmeren en besturen van een servomotor, zodat de draairichting kan worden geregeld door simpelweg op een knop te drukken. Dit project biedt niet alleen een uitstekende introductie in de wereld van motorbesturing, maar laat ook zien hoe je interacties kunt versterken door middel van visuele feedback op een TFT-scherm.

SERVOMOTOR: Een servo bestaat uit een elektromotor met tandwielkast en besturingselektronica. Aan de uitgaande kant van de tandwielkast zit een tandwiel waarop het servowiel is gemonteerd. Servo's worden gebruikt in de modelbouw, bijvoorbeeld om de vleugel- of roerstand van een vliegtuig of schip te regelen. Ook in de autotechniek worden steeds meer servo's gebruikt voor het automatisch sluiten van deuren, voor raamregelaars, spiegels en andere verstelbare elementen.



Sluit eerst de servomotor aan op de servo-interface met nummer 1 op je Explorer Board.



ATTENTIE! Voor dit project is het noodzakelijk om de schakelaar voor het TFT-scherm, de knoppen en de servo's op ON te zetten.

SAMENVATTING: We besturen onze servomotor en laten hem schakelen tussen links- en rechtsom draaien, bestuurd door onze knoppen.

Initialisatie van de servomotor en knoppen

```
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
```

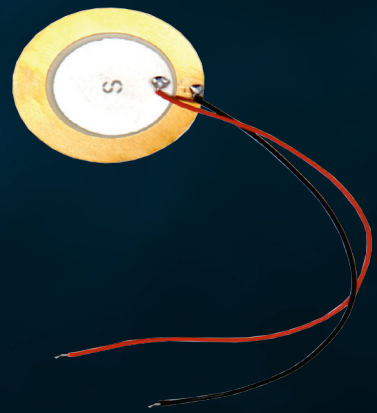
De knoppen controleren

```
    # 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
```

Servobesturing

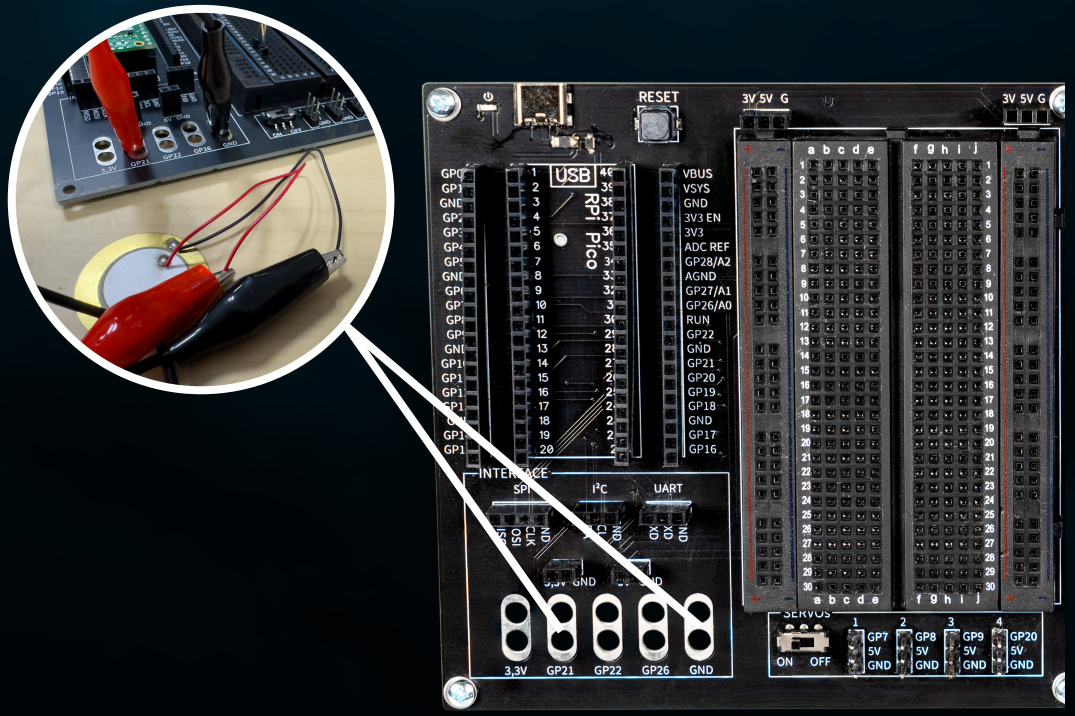
5.4 ZELFGEMAAKTE ZOEMER

Het vierde project in ons elektronica-avontuur met de Explorer Set draait om geluid! We duiken in de wereld van akoestische signalen door ons eigen zoemerschakeling te maken. Dit project geeft je niet alleen de kans om de basisprincipes van het maken van schakelingen te begrijpen, maar ook om te leren hoe je geluidssignalen kunt maken met eenvoudig gereedschap. Het gebruik van krokodillenklemmen maakt de bouw bijzonder gebruiksvriendelijk en toegankelijk, zelfs voor degenen die nog aan het begin van hun elektronica-reis staan.



Eerst sluiten we de zoemer voorzichtig aan op het Explorer Board met krokodillenklemmen. Deze klemmen zijn ideaal voor snelle en flexibele verbindingen zonder te hoeven solderen. De zoemer, een klein onderdeel dat geluid kan produceren, wordt het middelpunt van ons project. Wanneer er stroom op wordt gezet, gaat de zoemer trillen en produceert hij een geluid.

Sluit eerst een krokodillenklem aan op de GP21-krokodillenklemconnector op je Explorer-bord. Sluit het andere uiteinde van de krokodillenklem aan op de rode zoemerkabel. Sluit een andere krokodillenklem aan op de GND-krokodillenklemverbinding op uw Explorer-bord. Sluit het andere uiteinde van de klem aan op de zwarte zoemerkabel.



RASPBERRY PI PICO	ZOEMER
GP21	Rode kabel
GND	Zwarte kabel

ATTENTIE! Voor dit project is het nodig om de schakelaar voor de knoppen op ON te zetten.

SAMENVATTING: We sluiten een externe zoemer aan op onze Raspberry Pi Pico om een eenvoudig, leuk deuntje af te spelen.

```
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
```

Lijst met aantekeningen

Melodie geheugen

Functie voor het spelen van de noten

5.5 JE EIGEN CIRCUIT

In het vijfde project van ons elektronica-avontuur met de Explorer Set verkennen we de fascinerende wereld van lichtregeling. Deze keer bouwen we een schakeling waarmee je LED's kunt aansturen. Dit is een fantastische kans om de principes van elektronische schakelingen te begrijpen en tegelijkertijd controle te krijgen over de lichteffecten.

LEDS: LED's, of light-emitting diodes, zijn kleine maar krachtige lichtbronnen die in veel elektronische projecten worden gebruikt. Ze hebben veel voordelen ten opzichte van traditionele gloeilampen, zoals een langere levensduur, lager energieverbruik en de mogelijkheid om in verschillende kleuren op te lichten. Een LED bestaat uit een halfgeleidermateriaal dat licht uitstraalt wanneer er een elektrische stroom doorheen stroomt.

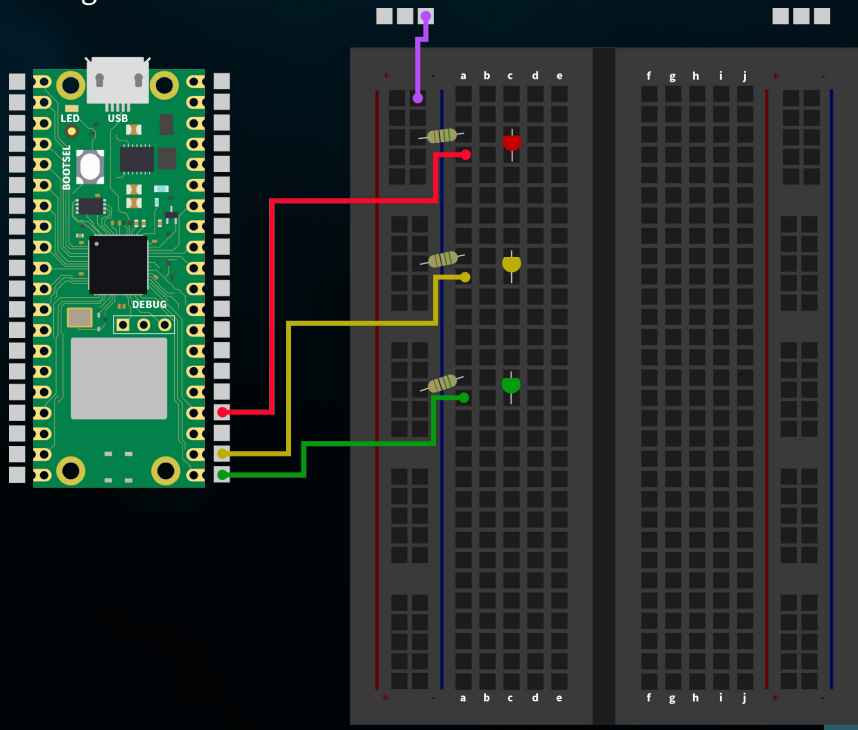
Het is belangrijk om de juiste polariteit van LED's te herkennen, omdat ze alleen werken als de stroom er in de juiste richting doorheen loopt. Dit betekent dat de positieve pool van de stroombron moet worden aangesloten op de positieve kant van de LED en de negatieve pool van de stroombron moet worden aangesloten op de negatieve kant van de LED.

Zo herken je de polariteit van een LED:

Langere poot: bij de meeste LED's is de langere poot de anode (+), dus de positieve aansluiting. Het kortere been is de kathode (-), d.w.z. de negatieve aansluiting.

Platte rand: Er kan een platte rand aan de zijkant van de LED-behuizing zitten. Deze kant markeert meestal de kathode, d.w.z. de negatieve pool.

Bouw eerst de schakeling na zoals weergegeven in het volgende schema. Maar zorg ervoor dat de polariteit van de LED's juist is. Gebruik ook een 56 Ω (groen-blauw-zwart) serieweerstand voor elke LED.



RASPBERRY PI PICO	LEDS
GP18	Rode LED
GP17	Gele LED
GP16	Groene LED

ATTENTIE! Voor dit project is het noodzakelijk om de schakelaar voor het TFT-scherm op OFF te zetten.

SAMENVATTING: We sturen drie verschillende LED's aan via de pinnen van de Raspberry Pi Pico, waarbij elke LED afwisselend knippert.

Initialisatie van de LED's

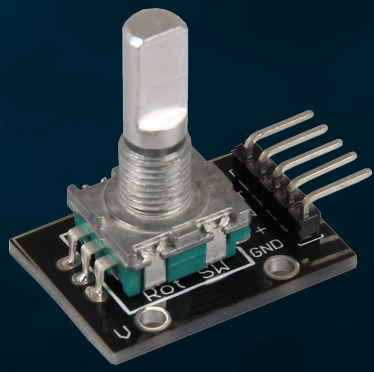
LED knipperfunctie

Hoofdlus

```
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
```

5.6 LED-BEDIENING

In het zesde project van ons elektronica-avontuur gebruiken we onze roterende encoder om de helderheid en kleur van LED's te regelen - een eenvoudige maar fascinerende manier om jezelf onder te dompelen in elektronica. De roterende encoder, ons centrale bedieningselement, maakt speelse interactie mogelijk dankzij zijn dubbele functie. Veranderingen in helderheid worden geregeld door te draaien, terwijl je door op de encoder te drukken door de kleuren van de LED's bladert - ideaal om de basisprincipes van elektronica en kleurmenging te illustreren.



ROTERENDE ENCODER: De roterende encoder is een slim apparaatje dat je draaibewegingen omzet in elektronische signalen. Stel je een draaiknop voor zoals je die kent van een radio. Wanneer je aan deze knop draait, kan de roterende encoder meten hoe ver en in welke richting je hem hebt gedraaid. Deze informatie kan vervolgens worden gebruikt om bijvoorbeeld het volume te wijzigen, door menu's te navigeren of, in onze projecten, de helderheid van LED's aan te passen.

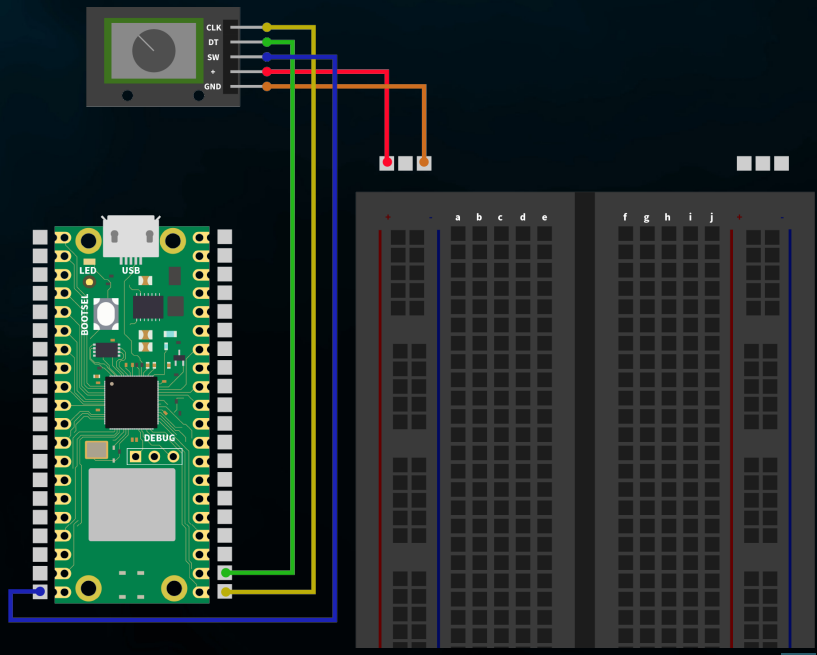
Roterende encoders hebben vaak ook een ingebouwde knop - dit betekent dat ze ook als drukschakelaar kunnen fungeren. Wanneer je op de draaiknop drukt, herkent de roterende encoder deze druk als een afzonderlijk signaal. Dit kan worden gebruikt voor verschillende functies, zoals het in- en uitschakelen van een apparaat of het wijzigen van de bedrijfsmodus.

SAMENGEVAT: Met een roterende encoder kun je verschillende commando's naar je elektronicaprojecten sturen door te draaien en te drukken. Het is een intuïtief en veelzijdig hulpmiddel dat interactie met je projecten gemakkelijk en leuk maakt.

Sluit eerst de roterende encoder als volgt aan op je Raspberry Pi Pico:

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

ATTENTIE! Voor dit project moet je de schakelaar voor het TFT-scherm op OFF zetten en de schakelaar voor de LED's op ON.



SAMENVATTING: We gebruiken de draaiknop om de kleur en helderheid van onze vier LED's te regelen. Door aan de encoder te draaien, veranderen we de helderheid en door op de encoder te drukken, passen we de kleur van de LED's aan.

```
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()
```

Initialisatie van de LED's en de draaiknop

Functie voor de roterende encoder

```
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)
```

Lijst met aantekeningen

5.7 AUTOMATISCHE HELDERHEIDSREGELING

In het zevende project van ons elektronica-avontuur gebruiken we een fotodiode om de helderheid van LED's automatisch te regelen. De fotodiode zet licht om in een elektrisch signaal zodat de LED's feller oplichten als het donker is en dimmen als er meer omgevingslicht is.

Door de fotodiode aan te sluiten op het Explorer Board en te programmeren op de Raspberry Pi Pico, passen de LED's zich intelligent aan de helderheid van de omgeving aan. Dit project laat zien hoe reactieve en energiebesparende elektronische systemen kunnen worden gebouwd met eenvoudige onderdelen.

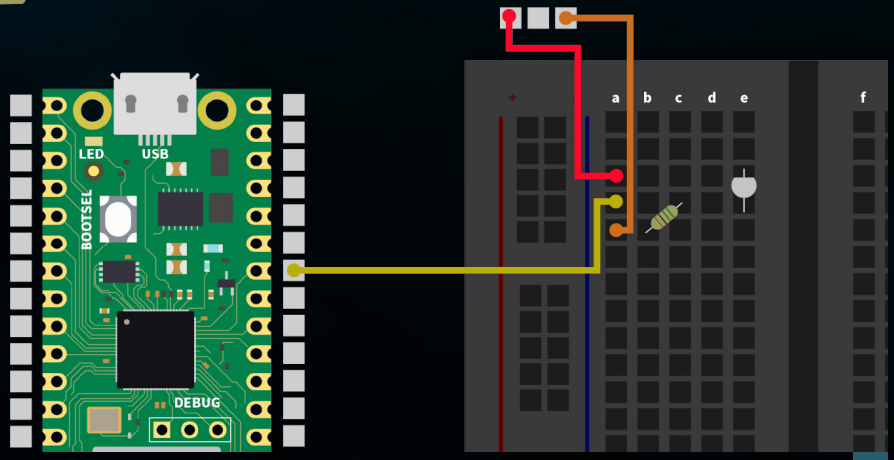
FOTODIODE: Een fotodiode is een speciaal type halfgeleider dat reageert op licht dat erop valt door een elektrische stroom te genereren. Zie een fotodiode als een klein zonnepaneel: wanneer er licht op valt, zet de fotodiode dit licht om in een elektrisch signaal. Hoe meer licht de fotodiode bereikt, hoe sterker het signaal wordt.

Fotodiodes zijn erg gevoelig en kunnen zelfs kleine hoeveelheden licht detecteren, waardoor ze ideaal zijn voor projecten waarbij het belangrijk is om de helderheid of aanwezigheid van licht te meten. Ze kunnen bijvoorbeeld worden gebruikt in automatische helderheidsregelaars, lichtsensoren of als onderdeel van een verlichtingscontrolesysteem.

Kortom, fotodiodes zijn effectieve lichtdetectoren waarmee we elektronische apparaten intelligent kunnen laten reageren op veranderingen in de omgevingsverlichting.

Sluit eerst de fotodiode als volgt aan via het breadboard. Houd er rekening mee dat ook hier het gebruik van een weerstand vereist is. Gebruik hier de weerstand van 100 kΩ (bruin-zwart-geel).

RASPBERRY PI PICO	FOTODIODE
3V3	Positieve kathode
GP28/A2	Negatieve anode



ATTENTIE! Voor dit project is het nodig om de schakelaar voor het relais op OFF te zetten en de schakelaar voor de LED's op ON.

SAMENVATTING: We gebruiken onze fotodiode om de helderheid van de omgeving te meten en de helderheid van vier LED's aan te passen. De intensiteit van de LED's verandert afhankelijk van het licht dat door de fotodiode wordt gedetecteerd, waarbij een donkere omgeving leidt tot fellere LED's en omgekeerd. Je kunt het beste een zaklamp gebruiken voor het best mogelijke resultaat.

```
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)
```

Initialisatie van de LED's en de fotodiode

Meting van de fotodiode en regeling van de helderheid

5.8 RGB LED-BEDIENING

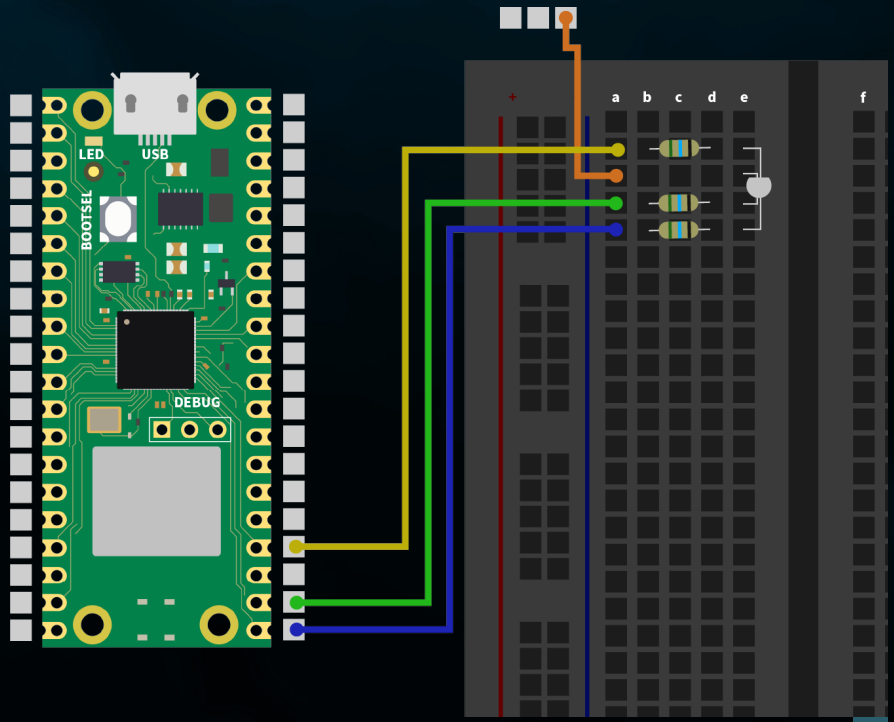
In het achtste en laatste project in onze elektronicaserie richten we ons op de kleurregeling van RGB-LED's met behulp van de geïntegreerde knoppen op het Explorer Board. RGB LED's zijn speciale lichtgevende diodes die rood, groen en blauw (RGB) licht combineren om een breed scala aan kleuren weer te geven. Door de intensiteit van elke kleurcomponent afzonderlijk aan te passen, kunnen we bijna elke kleur creëren.

In dit project sluiten we de RGB LED aan op het breadboard en gebruiken we de bestaande knoppen om de kleuren van de LED te regelen. Elke knop is toegewezen aan een kleur (rood, groen, blauw).

RGB LED: Een RGB LED combineert rood, groen en blauw in één lichtpunt. Door de helderheid van elk van de drie kleuren te veranderen, kan bijna elke kleur worden gecreëerd. Dit wordt gedaan door pulsbreedtemodulatie (PWM), die de intensiteit van elke kleur regelt. RGB LED's met slechts drie kleuren maken dus een breed kleurenspectrum mogelijk, ideaal voor kleurrijke verlichtingsprojecten.

Sluit eerst de RGB LED als volgt aan op het breadboard. Houd er rekening mee dat elk van de drie kleurkanalen hier ook een serieweerstand nodig heeft. Je moet hier de 56 Ω weerstand (groen-blauw-zwart) gebruiken.

RASPBERRY PI PICO	RGB-LED
GP18	Eerste pin
GND	Tweede pin
GP17	Derde pin
GP16	Vierde pin



ATTENTION! Voor dit project is het nodig om de schakelaar voor de TFT op OFF te zetten en die voor de knoppen op ON.

SAMENVATTING: De drie kleurkanalen van de RGB LED (rood, groen en blauw) worden in- en uitgeschakeld met de knoppen (links, boven en rechts).

```
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
```

Initialisatie van de LED en de knoppen

De knoppen testen en de LED aansturen

6. INFORMATIE & TERUGNAMEVERPLICHTINGEN

ONZE INFORMATIE- EN TERUGNAMEVERPLICHTINGEN VOLGENS DE DUITSE WET OP ELEKTRISCHE EN ELEKTRONISCHE APPARATUUR (ELEKTROG)

SYMBOOL OP ELEKTRISCHE EN ELEKTRONISCHE APPARATUUR: 
Deze doorgestreepte vuilnisbak betekent dat elektrische en elektronische apparaten niet bij het huishoudelijk afval horen. Je moet de oude apparaten inleveren bij een inzamelpunt. Voordat u ze inlevert, moet u gebruikte batterijen en accu's die niet bij het oude apparaat horen, scheiden.

RETOURMOGELIJKHEDEN:
Als eindgebruiker kunt u uw oude apparaat (dat in wezen dezelfde functie vervult als het nieuwe apparaat dat u bij ons koopt) gratis inleveren voor verwijdering wanneer u een nieuw apparaat koopt. Kleine apparaten met geen buitenafmetingen groter dan 25 cm kunnen worden weggegooid in normale huishoudelijke hoeveelheden, ongeacht of u een nieuw apparaat hebt gekocht.

MOGELIJKHEID TOT RETOURNEREN OP ONZE BEDRIJFSLOCATIE TIJDENS OPENINGSTIJDEN:
SIMAC Electronics GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

RETOURMOGELIJKHEID IN UW REGIO:
Wij sturen u een pakketzegel waarmee u het apparaat gratis naar ons kunt terugsturen. Neem hiervoor contact met ons op per e-mail op service@joy-it.net of per telefoon.

VERPAKKINGSINFORMATIE:
Verpak je oude apparaat goed voor transport. Als u geen geschikt verpakkingsmateriaal hebt of uw eigen verpakkingsmateriaal niet wilt gebruiken, neem dan contact met ons op en wij sturen u een geschikte verpakking.

7. SUPPORT

Ook na je aankoop staan we voor je klaar. Als er vragen onbeantwoord blijven of zich problemen voordoen, zijn we ook bereikbaar via e-mail, telefoon en het ticketondersteuningssysteem.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Telefoon: +49 (0)2845 9360 – 50 (ma. - do.: 09:00 - 17:00 of klok, vr: 09:00 - 14:30 of klok)

Ga voor meer informatie naar onze website:

WWW.JOY-IT.NET