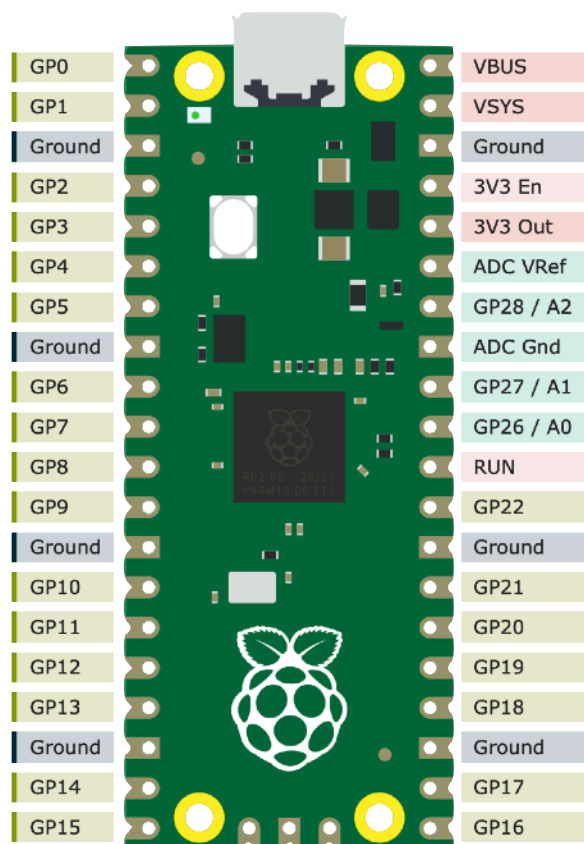


Programmeren met de Raspberry Pi Pico

Een handleiding voor kunstenaars en makers

De Raspberry Pi Pico is een **microcontroller**, wat betekent dat het een heel eenvoudige computer is. Je kunt er inputs op aansluiten, zoals lichtsensoren, en outputs zoals motoren en lampen.

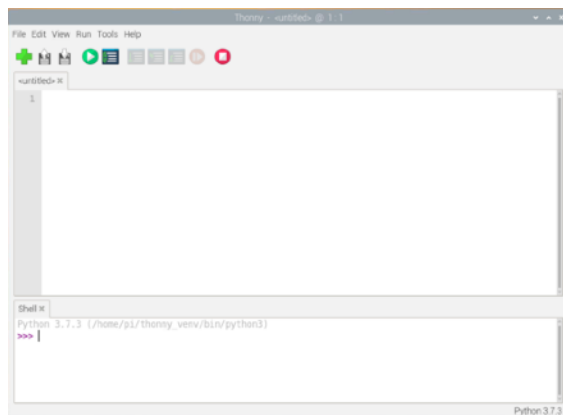
Dit schema toont de 'pinout' van de Pico. Het vertelt je welke pinnen je kunt gebruiken en wat hun nummers zijn.



Software installeren

Thonny is een programma op je computer waarmee je kunt programmeren in de programmeertaal Python.

Download het op thonny.org: kies de juiste versie voor jouw computer door in de rechterbovenhoek Windows, Mac of Linux te selecteren. Download het bestand, installeer het en open vervolgens het programma.



Typ deze regel in het bovenste venster van Thonny:

```
print('Hallo wereld!')
```

Klik vervolgens op de groene afspreekknop:



Nu wordt de code uitgevoerd. Het resultaat verschijnt in het onderste Thonny-venster.

Je eerste Pico-programma



Nu gaan we met behulp van Thonny code op het Pico-bordje uitvoeren in plaats van op je computer.

Sluit de Pico aan op je computer via een USB-kabel.

Selecteer 'MicroPython (Raspberry Pi Pico)' in de rechteronderhoek van het Thonny-scherm. (Als je die optie niet ziet in de lijst, controleer dan of je Raspberry Pi Pico goed is aangesloten.)

Wanneer je nu code uitvoert, zal het op de Pico worden uitgevoerd. Kopieer deze code naar Thonny en klik op de groene afspelenknop:

```
from machine import Pin
led = Pin(25, Pin.OUT)
led.toggle()
```



Elke keer dat je het script uitvoert (door op de groene knop te klikken), zie je het LED-lampje op de Pico aan- of uitgaan.

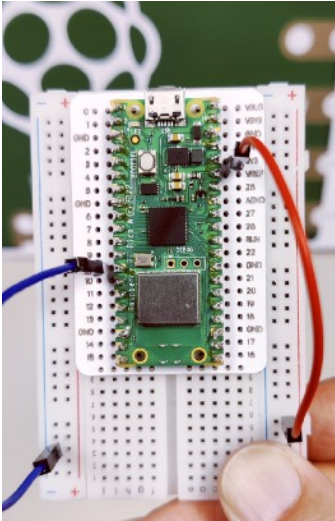
Je kunt ook de LED steeds laten knipperen zonder te wachten tot je het programma opnieuw uitvoert. Voeg deze regels toe aan de vorige code en voer het uit:

```
from time import sleep_ms
while True:
    led.toggle()
    sleep_ms(1000)
```

Zorg ervoor dat je **ook de witruimte** overneemt (4 spaties), want dat is belangrijk in Python.

LED aansluiten via een breadboard

Het aansluiten van sensoren en andere onderdelen op de Pico is vaak makkelijker met behulp van een breadboard.

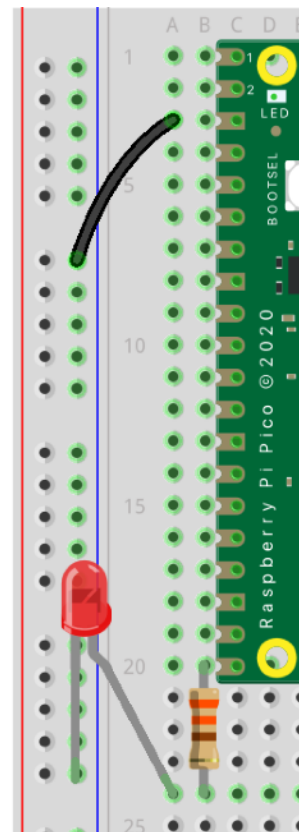


Je kunt draden en componenten in het breadboard steken. Gaatjes op dezelfde rij zijn met elkaar verbonden. Zo kun je dus onderdelen aansluiten op pinnen van de Pico.

De gaatjes naast elke gekleurde lijn aan de zijkanten van het breadboard zijn allemaal met elkaar verbonden. We gebruiken die kolommen om stroom te verdelen: de gaatjes naast de rode lijnen voor plus en de blauwe lijnen voor min.

Via het breadboard kun je bijvoorbeeld een LED-lampje aansluiten op de Pico.

1. Neem een **weerstand** van 330 Ohm en steek een van de pootjes in het breadboard naast pin GP15.
2. Steek het andere pootje in een van de rijen eronder.
3. De pootjes van LEDs zijn niet allebei even lang. Steek een LED met zijn langere pootje in dezelfde rij als de weerstand, zoals in het schema.
4. Steek het andere pootje in een gaatje van de blauwe min-kolom van het breadboard.
5. Gebruik een draad (bij voorkeur een zwarte) om de min-kolom te verbinden met een aarde-pin van de Pico ('GND', bijvoorbeeld de derde van boven óf de derde van onderen).



Verander in het script van net het pinnummer van 25 naar 15.

Als je nu het script aanzet dan zou de LED op het breadboard moeten knipperen!

Oefening

1. Kun je de LED tien keer sneller laten knipperen? (Tip: de wachttijd tussen aan en uit staat in de `sleep_ms`-functie in milliseconden.)

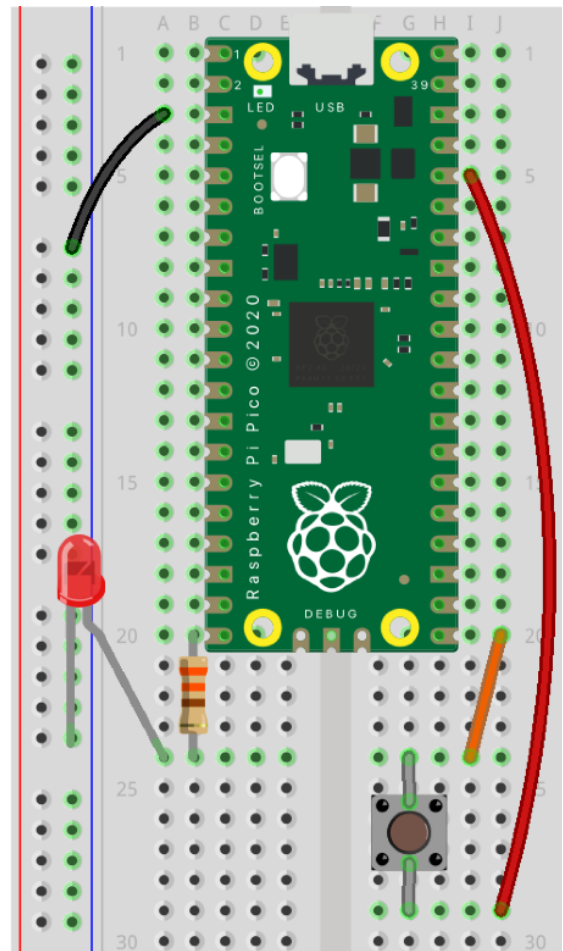
2. Kun je de LED verschillende patronen laten knipperen? Dus een paar keer korter en langer aan en uit? Je kunt de LED aanzetten met `led.on()` en uitzetten met `led.off()`.

Digitale sensoren: een schakelaar

We kunnen sensoren aansluiten op de Pico om hem te laten reageren op zijn omgeving. Een schakelaar is de aller-eenvoudigste sensor, dus laten we daarmee beginnen.

Sluit een drukknop aan op de Pico zoals in het diagram:

- Plaats de knop in het breadboard, met elk pootje in een andere rij.
- Verbind één poot van de knop via een draadje met de 3.3V-pin van de Pico (5e aan de rechterkant).
- Verbind de andere kant van de knop met pin GP16.




Kopieer en plak de volgende code in Thonny:

```
from machine import Pin
import time

led = Pin(15, Pin.OUT)
button = Pin(16, Pin.IN, Pin.PULL_DOWN)

while True:
    if button.value():
        led.toggle()
        time.sleep(0.5)
```

Druk op de afspeelknop  om het script uit te voeren.

 Druk nu de drukknop op het breadboard in!

Je zou nu je eerste werkende digitale sensor moeten hebben. Wanneer je op de knop drukt, gaat de LED op de Pico aan. Dus nu combineer je al een sensor met een output!

If-statements

Om ervoor te zorgen dat de Pico de LED alleen aan of uit zet wanneer er iets verandert, kunnen we **if-statements** gebruiken.

In het script hierboven zit bijvoorbeeld een if-statement:

```
if button.value():  
    led.toggle()  
    time.sleep(0.5)
```

Hier is de vraag: 'wordt de knop ingedrukt?' Als het antwoord 'ja' is ('True'), wordt de LED aan- of uitgezet.

Je kunt het ook zo veranderen dat de LED aangaat wanneer de knop wordt ingedrukt en uitgaat wanneer dat niet zo is:

```
if button.value():  
    led.on()  
else:  
    led.off()
```

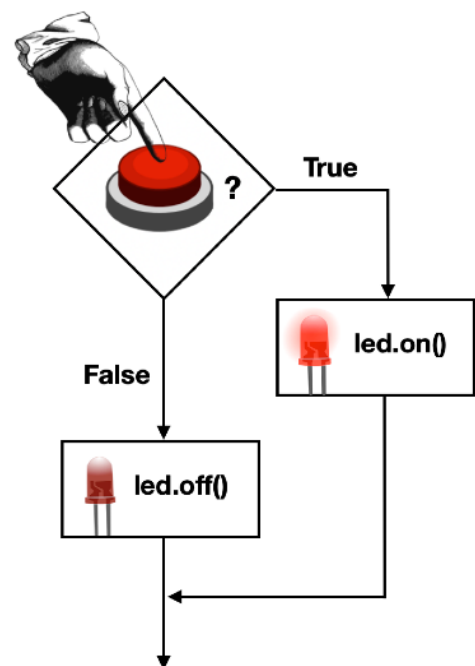
Oefening

Met de zojuist genoemde code gaat de LED aan wanneer je op de knop drukt.

Kun je het script aanpassen zodat de LED *uitgaat* wanneer je op de knop drukt?


Iets steeds opnieuw doen

Dus als de knop is ingedrukt en `button.value()` 'True' retourneert, schakelt de Pico de LED aan of uit en wacht daarna een halve seconde:



```
while True:
    if button.value(): # ga na of de knop wordt ingedrukt
        led.toggle() # zet de LED aan of uit
        time.sleep(0.5) # wacht een halve seconde
```

Dit gebeurt allemaal binnen een **while-lus**. Wat er in de while-lus staat, wordt steeds opnieuw gedaan. Wat bij de while-lus hoort, kun je zien aan de spaties voor de regels.

Op deze manier wordt hier de knop dus continu in de gaten gehouden, totdat je op de stopknop van Thonny drukt. 

Analoge sensor: lichtgevoelige weerstand

In tegenstelling tot schakelaars en andere digitale sensoren, die alleen 1 en 0 kennen (aan of uit, waar of onwaar), kunnen analoge sensoren tussenliggende waarden geven.

Een voorbeeld is de lichtsensor, die kan waarnemen of het heel licht is of helemaal donker, maar ook alles daartussen.



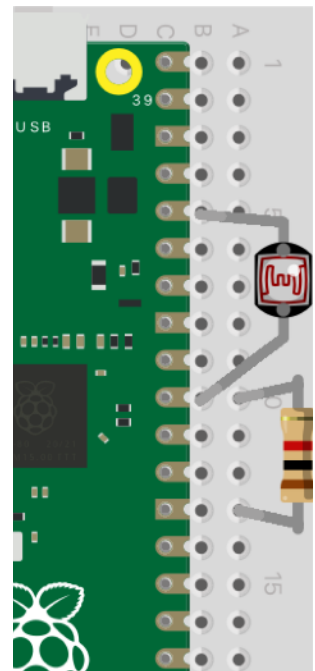
Om die lichtwaarden te kunnen lezen met de Pico, sluit je een lichtsensor aan op 3.3V (5e pin van boven aan de rechterkant) en op pin GP26 (10e pin).

Sluit vervolgens een weerstand aan met een waarde tussen 500 en 1.000 Ohm op GP26 en op GND.

Voer nu dit script uit:

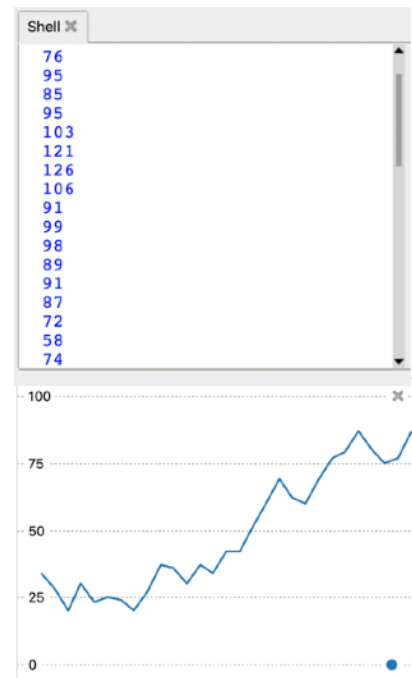
```
import machine, time
light_sensor = machine.ADC(26)

while True:
    print(light_sensor.read_u16())
    time.sleep_ms(70)
```



De print() functie zal constant data van de lichtsensoren naar je computer sturen. Dat bericht verschijnt in het shell-venster. (Zie je het shell-venster niet? Maak het zichtbaar via het 'View'-menu van Thonny.)

Om waarden op een meer visuele manier weer te geven, heeft Thonny ook een ingebouwde **plotter**. In dat venster zou je een grafiek van de veranderende licht-waarden moeten zien!



Oefening

Er zijn heel veel andere manieren waarop je de data van de sensor kunt gebruiken. Je zou bijvoorbeeld de LED sneller of langzamer kunnen laten knipperen door de lichtsensoren te bedekken of er juist een felle lamp op te laten schijnen!

Probeer het maar eens met de LED van daarstraks. Je moet een combinatie maken van beide scripts. In plaats van de waarde van de sensor alleen maar uit te printen, gebruik je die waarde nu ook als de wachttijd tussen het aan- en uitschakelen:

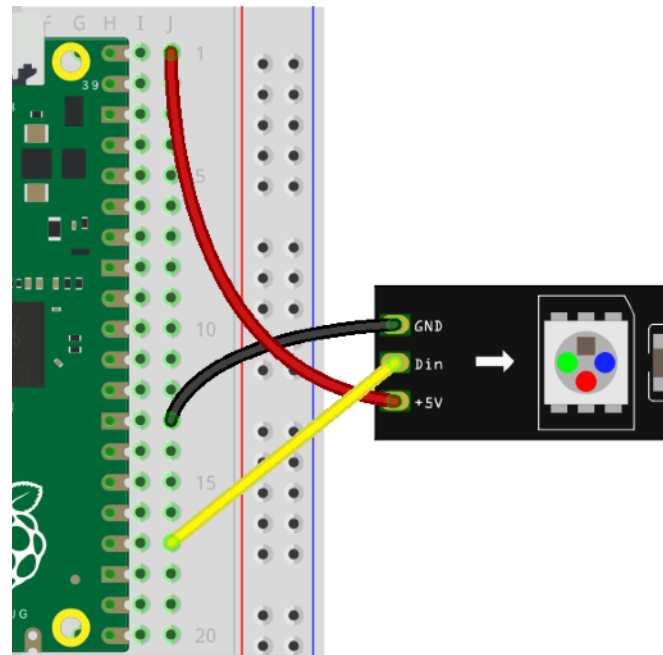
```
waarde = light_sensor.read_u16()/65535
print(waarde)
led.toggle()
time.sleep(waarde)
```

LED-strips met Pico

LEDs worden gebruikt in alles, van fietslampjes tot televisies. Je hebt al enkele LEDs met de Pico gebruikt; hier zullen we een langere strip LEDs aansluiten.

Verbind de LED-strip zoals aangegeven in het diagram.

1. '5V' op de LED-strip gaat naar VBUS, de eerste pin rechtsboven op de Pico.
2. 'GND' gaat naar één van de aarde-pins op de Pico.
3. Sluit 'DIN' aan op GP18 (4e pin van rechtsonder).



Als je nu deze code uitvoert, dan zou je licht moeten zien!

```
import machine, neopixel, time

leds_pin = machine.Pin(18)
aantal_leds = 15

np = neopixel.NeoPixel(leds_pin, aantal_leds)

np[0] = (255, 0, 0) # stelt de eerste LED in op rood

np.write() # update de LED strip
```

Je ziet dat de LEDs genummerd zijn, te beginnen met nummer 0. De kleuren geef je op in de volgorde rood, groen, blauw. Als je een kleur instelt op 0 dan is hij helemaal donker, en 255 is het felst. Een waarde van 128 is dus halve helderheid.

Probeer maar eens een paar andere LEDs een andere kleur te geven!

Stel dat je de hele strip dezelfde kleur wil geven, gebruik dan `fill()`:


```
kleur = (0,0,20) # eerst een kleur definiëren
np.fill(kleur) # alle leds dezelfde kleur geven
np.write() # laat de veranderingen zien op de strip
```

Met wat slimme lussen (for en while) kun je nu alle LEDs langsgaan om ze elk een andere kleur te geven. Op deze manier bijvoorbeeld kun je ze één voor één aan laten gaan:

```
while True: # doe wat nu volgt de hele tijd

    for i in range(aantal_leds): # ga alle LEDs één voor één langs

        np.fill((0, 0, 0)) # zet alle LEDs op zwart (dus uit)
        np[i] = (0, 128, 128) # zet deze LED aan
        np.write() # laat het zien op de strip

        time.sleep_ms(150) # laat hem even aan voordat we doorgaan
```

Oefening

1. Kun je de LEDs allemaal tegelijk laten knipperen? Gebruik fill() en de while-lus helemaal bovenaan deze handleiding.
2. Hierboven heb je gelezen over if-statements. Probeer nu de LED-strip alleen te laten oplichten als de knop is ingedrukt óf als de waarde van de lichtgevoelige sensor lager is dan een bepaald getal!
3. Maak de de kleuren afhankelijk van de waarde van de lichtgevoelige sensor. Dus bijvoorbeeld: hoe donkerder het is, des te meer rood er te zien is.

Geluiden maken met buzzers 🎵 🔊

Je kunt met de Pico ook geluiden maken en zelfs muziek maken.

Verbind de buzzer met een aarde-pin en pin 20 van de Pico.

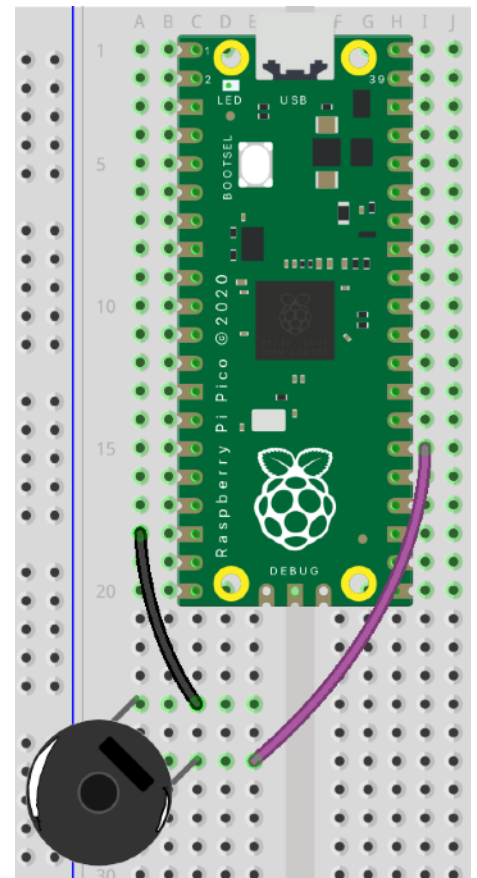
Je moet de buzzer misschien wat schuin in het breadboard zetten om hem te laten passen.

Kopiëer deze code naar Thonny (de uitleg in rood hoef je niet over te nemen):

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

buzzer = PWM(Pin(20)) # zet PWM aan op pin 20

buzzer.freq(400)      # toonhoogte
buzzer.duty_u16(1000) # volume (0-1000)
sleep(1)              # wacht 1 seconde
buzzer.duty_u16(0)    # zet geluid uit
```



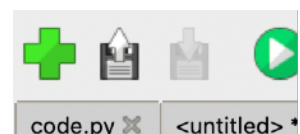
Als je iets op de buzzer legt, bijvoorbeeld een stukje karton of je vinger, dan klinkt hij iets luider.

Oefeningen

1. Verander zelf een paar keer de toonhoogte en de duur van de toon.
2. Maak zelf een kort muziekje door verschillende tonen achter elkaar af te spelen.

Muziek

We hebben ook al een paar muziekjes klaargezet. De scripts hiervoor staan al op je Pico. Klik in Thonny op het tweede icoontje van links om die te openen:



Thonny vraagt je nu of je een bestand wil openen van je computer of van de Pico. Kies 'Raspberry Pi Pico' en klik in het volgende venster op 'play.py' en dan op 'OK'.

Je kunt deze code nu uitvoeren om een muziekje te horen.

Je kunt het nummer in de variabele 'track' (op de negende regel) veranderen in een ander nummer om een ander deuntje te kiezen. De hele lijst zie je in het bestand 'melodies.py' dat ook op je Pico staat.

Chiptune

Chiptune, of 8-bit muziek, is een muziekgenre van muziek die gemaakt is met computers of geluidschips. Veel mensen kennen het als de muziek uit retro-computerspellen.

Je kunt interessante én irritante geluidseffecten maken door geluiden te programmeren, bijvoorbeeld door verschillende tonen heel kort en snel achter elkaar af te spelen.

In dit voorbeeld wordt de toonhoogte snel verlaagd om een vogelgeluid te creëren:

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

buzzer = PWM(Pin(20)) # zet PWM aan op pin 20
buzzer.duty_u16(1000)

for _ in range(2): # doe wat volgt twee keer
    for i in range(5000, 2999, -100): # afnemende frequentie
        buzzer.freq(i)
        sleep_ms(20) # zeer korte duur
        sleep_ms(200)

buzzer.duty_u16(0000) # zet de buzzer uit
```

In dit voorbeeld gaat de frequentie geleidelijk omhoog om een positief geluid te creëren:

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

buzzer = PWM(Pin(20)) # zet PWM aan op pin 20
buzzer.duty_u16(1000)

for i in range(2000, 5000, 100): # afnemende frequentie
    buzzer.freq(i)
    sleep_ms(50) # heel kort

buzzer.duty_u16(000) # zet de buzzer uit
```

Probeer ook deze code om steeds van hoge tonen naar lage tonen te gaan:

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

up = True
frequency = 88

buzzer = PWM(Pin(20))
buzzer.duty_u16(1000) # geluidssterkte

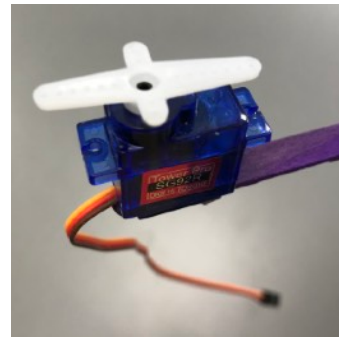
try:
    while True:
        buzzer.freq(frequency)
        sleep(0.1)
        if up:
            frequency = int(frequency * 1.1)
        else:
            frequency = int(frequency / 1.4)

        if frequency > 880:
            up = False
        elif frequency < 88:
            up = True
            frequency = 88
except KeyboardInterrupt:
    buzzer.duty_u16(0000) # zet de buzzer uit als het script stopt
```

Experimenteer nu zelf met het afspelen van korte noten en het veranderen van de frequentie in de **for-lussen** die je gezien hebt. Je kunt waarden gebruiken tussen 150 en 10000.

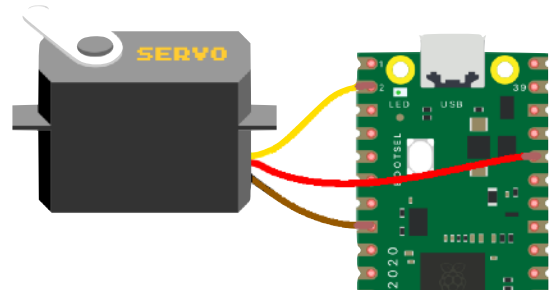
Dingen laten bewegen met servo-motoren

De motor aan de rechterkant wordt een microservo genoemd. Dit type motor vind je in veel speelgoed en robots. Ze zijn meestal beperkt tot 180 graden rotatie.



De meeste kleine servomotoren hebben drie draden:

- bruin is aarde (GND)
- rood is plus (dus voltage/3.3V)
- geel is signaal, om de servo te vertellen waarheen te draaien. Sluit hem aan op pin GP1.



Laat de servo heen en weer gaan tussen zijn minimum en maximum:

```
from piczero import Servo
servo = Servo(1)
servo.pulse()
```

Gebruik een **for-lus** om de servo heen en weer te laten bewegen in 100 stapjes:

```
from piczero import Servo
from time import sleep
servo = Servo(1)

for i in range(0, 100):
    servo.value = i / 100
    sleep(0.1)

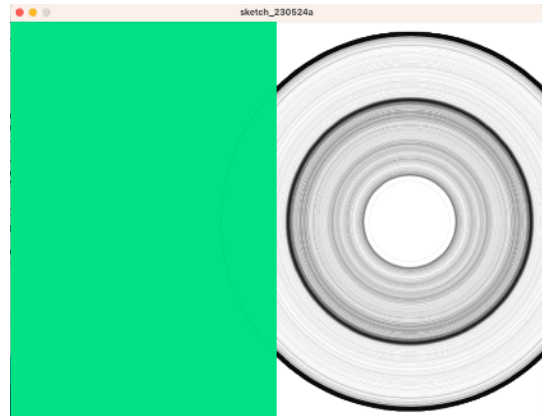
servo.off()
```

Oefeningen

1. Beweeg de servo een paar keer naar verschillende posities en laat hem tussendoor even wachten. Je hebt er maar een paar `servo.value()`'s en `sleep()`'s voor nodig.
2. Probeer ook `servo.min()`, `servo.mid()` en `servo.max()` om de servo naar zijn minimum-, midden- en maximumposities te laten bewegen.

Pico gebruiken met Processing

De Pico stuurt gegevens naar je computer via de USB-kabel, met behulp van `print()`. Die gegevens kunnen als invoer worden gebruikt voor veel verschillende programma's om interactieve muziek, games en animaties te maken.



Laten we het proberen met een sensor en Processing.

Zet de lichtsensoren van de vorige pagina op analoge sensoren op. Sla het eerste script op die pagina op de Pico op als 'main.py'. Kies vervolgens 'Send EOF / Soft reboot' uit het menu 'Run' om de Pico dat bestand te laten starten.

Download het Processing-programma van [Processing.org](https://processing.org).

Nadat het is geïnstalleerd, gebruik je het om de code hiernaast uit te voeren - je zou een animatie op je scherm moeten zien die reageert op veranderende waarden van de lichtsensoren.

Als er niets gebeurt, moet je waarschijnlijk het **port number** in de code wijzigen. Het script print een lijst met poorten op je computer naar het shell-venster onderaan. Kies het nummer van de poort waarmee de Pico is verbonden. (Het is zoiets als `/dev/ttyACM0` of `/dev/cu.usbmodem14201`, maar het verandert en hangt af van je besturingssysteem.)

Zorg er ook voor dat Thonny niet meer is verbonden met de Pico. Als dat wel zo is, geeft Processing een 'Port busy'-fout weer. In dat geval klik je op 'Disconnect' in het 'Run'-menu van Thonny.

```
import processing.serial.*;
Serial port;
void setup() {
  size(800, 600);
  printArray(Serial.list());

  // change port number here:
  port = new Serial(this, Serial.list()[1]);

  background(255, 10);
  colorMode(HSB, 65535);
}

void draw() {
  while (port.available() > 0) {
    String s = port.readStringUntil(10);
    if (s != null) {
      try {
        float intVal=Integer.parseInt(s.trim());
        println(intVal);
        fill(intVal/2, intVal, intVal);
        noStroke();
        rect(0, 0, width/2, height);
        noFill();
        stroke(0, 3000);
        circle(600, 300, (intVal/100));
      } catch (NumberFormatException npe) {
        // do nothing when incoming data
        // is not a whole number.
      }
    }
  }
}
```

Problemen oplossen

Foutmeldingen zijn je vrienden. Ze vertellen je wat er fout gaat en waar. Wanneer je een bericht zoals dit ziet, klik dan op de blauwe link. De regel `>>> |` waar het fout gaat wordt dan in de code zichtbaar, zodat je kunt zien waar je het probleem kunt oplossen.

```
Traceback (most recent call last):  
  File "<stdin>", line 8  
SyntaxError: invalid syntax
```

Problemen met verbinden

1. Selecteer de juiste interpreter rechtsonder in het Thonny-scherm. Als je wilt dat je code op de Pico wordt uitgevoerd, moet er 'MicroPython Raspberry Pi Pico' staan.
2. Is de USB-kabel correct aangesloten?
3. Probeer de kabel op een andere USB-poort aan te sluiten (ja, het klinkt raar maar soms helpt het).
4. Kun je een verbinding maken met de Pico wanneer er niets anders op is aangesloten?
5. Probeer een andere Pico.

Problemen met code

Vaak zijn de foutmeldingen die de Pico in de shell geeft echt nuttig. Probeer te begrijpen wat ze betekenen.

1. Klik op de link in de foutmelding om erachter te komen waar in de code de fout optreedt. Het zou de regel met het probleem moeten markeren.
2. **IndentationError: unexpected indent**: zorg ervoor dat regels binnen een if-statement of while-lus hetzelfde aantal spaties ervoor hebben.
3. **ModuleNotFoundError: No module named '..'**: installeer de vereiste bibliotheek.
4. Andere fouten: zoek online naar oplossingen voor de specifieke foutmeldingen die je ziet. Zoek met termen tussen aanhalingstekens, zoals 'Pico "exacte fout hier"'.

Geen fout, maar het werkt ook niet zoals verwacht?

- Controleer of alle componenten zijn aangesloten op de pinnen waaraan ze zijn toegewezen in de code.
- Gebruik `print()`'s om in de shell te kunnen zien wat de Pico aan het doen is.

- Als je er niet uitkomt: probeer code te vinden die doet wat je wilt online.
- Probeer een andere van de componenten die je gebruikt om erachter te komen of de eerste kapot zou kunnen zijn. Probeer daarna verschillende draden of zelfs een ander breadboard.

Verantwoording

Dit is een verkorte versie van de Pico-handleiding geschreven door Jaap Meijers. Deze versie voor Coderdojo werd gepubliceerd in januari 2024 onder deze licentie: Creative Commons Attribution 4.0 International-licence (<https://creativecommons.org/licenses/by/4.0/> © ⓘ). Dat betekent dat het je vrij staat om deze handleiding te verspreiden en aan te passen, zolang je aan bronvermelding doet, aangeeft of er wijzigingen zijn aangebracht en een link naar de licentie opneemt.

Beelden

- Pico-pinout-diagram: Raspberry Pi Foundation
- Emoticons: <https://freesvg.org>
- Alle overige diagrammen: Fritzing.org
- Lichtsensor (LDR): Arnau 944/Wikimedia