Recipes

The recipes provide examples of how you can use picozero.

Importing picozero

You will need add an *import* line to the top of your script to use picozero.

You can import just what you need, separating items with a comma `,`:

```
from picozero import pico_led, LED
```

Now you can use `pico_led` and `LED` in your script:

```
pico_led.on() # Turn on the LED on the Raspberry Pi Pico
led = LED(14) # Control an LED connected to pin GP14
led.on()
```

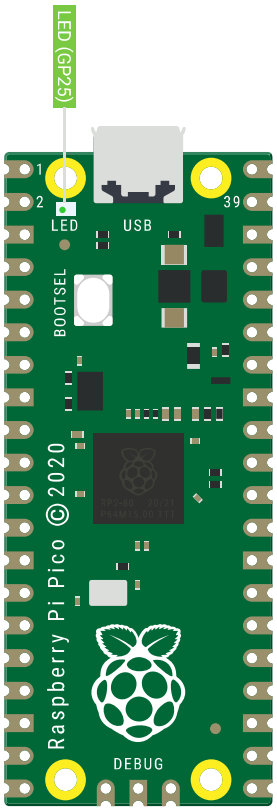
Alternatively, the whole picozero library can be imported:

```
import picozero
```

In this case, all references to picozero items must be prefixed:

```
picozero.pico_led.on()
led = picozero.LED(14)
```

Pico LED



To turn on the LED on your Raspberry Pi Pico:

```
from picozero import pico_led

pico_led.on()
```

Run your script to see the LED turn on.

Using the `pico_led` is equivalent to:

```
pico_led = LED(25)
```

You can use `pico_led` in the same way as external LEDs created using `LED`.

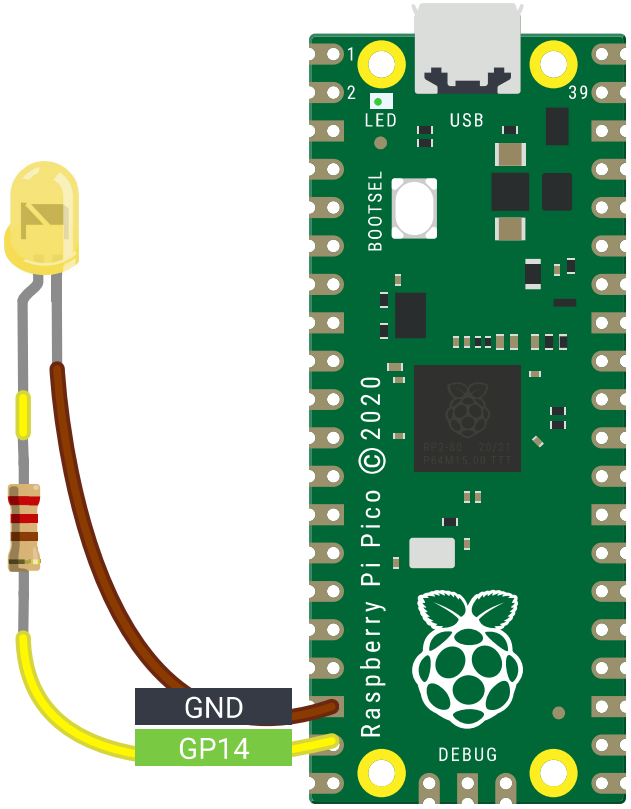
Pin out

You can output a *diagram* of the Raspberry Pi Pico which displays its pins and the

```
from picozero import pinout

pinout()
```

```
        ---usb---
GP0  1  |o     o| -1  VBUS
GP1  2  |o     o| -2  VSYS
GND  3  |o     o| -3  GND
GP2  4  |o     o| -4  3V3_EN
GP3  5  |o     o| -5  3V3(OUT)
GP4  6  |o     o| -6           ADC_VREF
GP5  7  |o     o| -7  GP28     ADC2
GND  8  |o     o| -8  GND      AGND
GP6  9  |o     o| -9  GP27     ADC1
GP7  10 |o     o| -10 GP26     ADC0
GP8  11 |o     o| -11 RUN
GP9  12 |o     o| -12 GP22
GND  13 |o     o| -13 GND
GP10 14 |o     o| -14 GP21
GP11 15 |o     o| -15 GP20
GP12 16 |o     o| -16 GP19
GP13 17 |o     o| -17 GP18
GND  18 |o     o| -18 GND
GP14 19 |o     o| -19 GP17
GP15 20 |o     o| -20 GP16
        ---------
```

LEDs



You can control external LEDs with a Raspberry Pi Pico.

Flash

Turn an `LED` on and off:

```
from picozero import LED
from time import sleep

led = LED(14)

led.on()
sleep(1)
led.off()
```

Toggle an `LED` to turn it from on to off or off to on:

```
from picozero import LED
from time import sleep

led = LED(14)

while True:
    led.toggle()
    sleep(1)
```

Alternatively, you can use the `blink()` method.

```
from picozero import LED

led = LED(14)

led.blink()
```

Brightness

Set the brightness of an `LED`:

```
from picozero import LED
from time import sleep

led = LED(14)

while True:
    led.brightness = 0  # off
    sleep(1)
    led.brightness = 0.5  # half brightness
    sleep(1)
    led.brightness = 1  # full brightness
    sleep(1)
```

Create a pulse effect:

```
from picozero import LED
from time import sleep
from math import sin, radians

led = LED(14)

while True:
    for i in range(360):
        angle = radians(i)
        led.brightness = 0.5 + 0.5 * sin(angle)
        sleep(0.01)
```

Alternatively, you can use the `pulse()` method.

```
from picozero import LED

led = LED(14)

led.pulse()
```

Buttons

You can connect buttons and switches to a Raspberry Pi Pico and detect when they are pressed.

Check if a `Button` is pressed:

```
from picozero import Button
from time import sleep

button = Button(18)

while True:
    if button.is_pressed:
        print("Button is pressed")
    else:
        print("Button is not pressed")
    sleep(0.1)
```

Run a function every time a `Button` is pressed:

```
from picozero import Button, pico_led
from time import sleep

button = Button(18)

def led_on_off():
    pico_led.on()
    sleep(1)
    pico_led.off()

button.when_pressed = led_on_off
```

Note

The line `button.when_pressed = led_on_off` does not run the function `led_on_off`, rather it creates a reference to the function to be called when the button is pressed. Accidental use of `button.when_pressed = led_on_off()` would set the `when_pressed` action to `None` (the return value of this function), which would mean nothing happens when the button is pressed.

Turn the `pico_led` on when a `Button` is pressed and off when it is released:

```
from picozero import Button, pico_led

button = Button(18)

button.when_pressed = pico_led.on
button.when_released = pico_led.off
```

RGB LEDs

Set colours with an `RGBLED`:

```
from picozero import RGBLED
from time import sleep

rgb = RGBLED(red=2, green=1, blue=0)

rgb.red = 255  # full red
sleep(1)
rgb.red = 128  # half red
sleep(1)

rgb.on() # white

rgb.color = (0, 255, 0)  # full green
sleep(1)
rgb.color = (255, 0, 255)  # magenta
sleep(1)
rgb.color = (255, 255, 0)  # yellow
sleep(1)
rgb.color = (0, 255, 255)  # cyan
sleep(1)
rgb.color = (255, 255, 255)  # white
sleep(1)

rgb.color = (0, 0, 0)  # off
sleep(1)

# slowly increase intensity of blue
for n in range(255):
    rgb.blue = n
    sleep(0.01)

rgb.off()
```

Use `toggle()` and `invert()`:

```
from picozero import RGBLED
from time import sleep

rgb = RGBLED(red=2, green=1, blue=0)

rgb.color = (255, 165, 0) # orange
sleep(1)

for _ in range(6):
    rgb.toggle()
    sleep(1)

for _ in range(6):
    rgb.invert()
    sleep(1)

rgb.off()
```

Blink

Use `blink()` blink to change between colours. You can control which colours are used and how long the LED is set to each colour. The colour *(0, 0, 0)* represents off.

You can control whether `blink()` runs a fixed number of times and whether it waits until it has finished or returns immediately so other code can run.

```
from picozero import RGBLED
from time import sleep

rgb = RGBLED(1, 2, 3)

rgb.blink() # does not wait
sleep(6)
rgb.off()
sleep(1)

# blink purple 2 seconds, off 0.5 seconds
rgb.blink(on_times=(2, 0.5), colors=((1, 0, 1), (0, 0, 0)), wait=True, n=3)

rgb.off()
sleep(1)

# blink red 1 second, green 0.5 seconds, blue 0.25 seconds
rgb.blink((1, 0.5, 0.25), colors=((1, 0, 0), (0, 1, 0), (0, 0, 1)), wait=True, n=2)
```

Pulse

Use `pulse()` to gradually change the LED colour. The default will pulse between red and off, then green and off, then blue and off.

```
from picozero import RGBLED
from time import sleep

rgb = RGBLED(1, 2, 3)

rgb.pulse() # does not wait
sleep(6)
rgb.off()
sleep(1)

# 2 second to fade from purple to off, 0.5 seconds to change from off to purple
rgb.pulse(fade_times=(2, 0.5), colors=((1, 0, 1), (0, 0, 0)), wait=True, n=3)

rgb.off()
sleep(1)

# 4 seconds to change from red to green, 2 to change from green to blue, then 1 to change from
blue back to red
rgb.pulse((4, 2, 1), colors=((1, 0, 0), (0, 1, 0), (0, 0, 1)), wait=True, n=2)
```

Cycle

The default for `cycle()` is to cycle from red to green, then green to blue, then blue to red.

```
from picozero import RGBLED
from time import sleep

rgb = RGBLED(1, 2, 3)

# Gradually colour cycle through colours between red and green, green and blue then blue and red
rgb.cycle()
sleep(4)
rgb.off()
sleep(1)

# Colour cycle slower in the opposite direction
rgb.cycle(fade_times=3, colors=((0, 0, 1), (0, 1, 0), (1, 0, 0)), wait=True, n=2)
rgb.off()
```

Potentiometer

Print the value, voltage, and percent reported by a potentiometer:

```
# Potentiometer connected to GP26 (ADC0), GND and 3V

from time import sleep
from pico import Pot

pot = Pot(26)

while True:
    print(pot.value, pot.voltage)
    sleep(0.1)
```

Note

In the Thonny Python editor, choose **View** > **Plotter** to plot the output of `print()`.

Use a potentiometer to control the brightness of an LED:

```
from picozero import Pot, LED

# Potentiometer connected to GP26 (ADC0), GND and 3V
# LED connected to GP0

pot = Pot(26)
led =  LED(0)

while True:
    led.value = pot.value
```

Buzzer

Control an active buzzer that plays a note when powered:

```
# Active Buzzer that plays a note when powered
from time import sleep
from picozero import Buzzer

buzzer = Buzzer(10)

buzzer.on()
sleep(1)
buzzer.off()
sleep(1)

buzzer.beep()
sleep(4)
buzzer.off()
```

Speaker

Control a passive buzzer or speaker that can play different tones or frequencies:

```
from picozero import Speaker
from time import sleep

speaker = Speaker(5)

def tada():
    c_note = 523
    speaker.play(c_note, 0.1)
    sleep(0.1)
    speaker.play(c_note, 0.9)

def chirp():
    global speaker
    for _ in range(5):
        for i in range(5000, 2999, -100):
          speaker.play(i, 0.01)
        sleep(0.2)


try:
    tada()
    sleep(1)
    chirp()

finally: # Turn the speaker off if interrupted
    speaker.off()
```

Play a tune

Play a tune of note names and durations in beats:

```
from picozero import Speaker

speaker = Speaker(5)

BEAT = 0.25 # 240 BPM

liten_mus = [ ['d5', BEAT / 2], ['d#5', BEAT / 2], ['f5', BEAT], ['d6', BEAT], ['a#5', BEAT],
['d5', BEAT],
              ['f5', BEAT], ['d#5', BEAT], ['d#5', BEAT], ['c5', BEAT / 2],['d5', BEAT / 2],
['d#5', BEAT],
              ['c6', BEAT], ['a5', BEAT], ['d5', BEAT], ['g5', BEAT], ['f5', BEAT], ['f5', BEAT],
['d5', BEAT / 2],
              ['d#5', BEAT / 2], ['f5', BEAT], ['g5', BEAT], ['a5', BEAT], ['a#5', BEAT], ['a5',
BEAT], ['g5', BEAT],
              ['g5', BEAT], ['', BEAT / 2], ['a#5', BEAT / 2], ['c6', BEAT / 2], ['d6', BEAT / 2],
['c6', BEAT / 2],
              ['a#5', BEAT / 2], ['a5', BEAT / 2], ['g5', BEAT / 2], ['a5', BEAT / 2], ['a#5',
BEAT / 2], ['c6', BEAT],
              ['f5', BEAT], ['f5', BEAT], ['f5', BEAT / 2], ['d#5', BEAT / 2], ['d5', BEAT],
['f5', BEAT], ['d6', BEAT],
              ['d6', BEAT / 2], ['c6', BEAT / 2], ['b5', BEAT], ['g5', BEAT], ['g5', BEAT], ['c6',
BEAT / 2],
              ['a#5', BEAT / 2], ['a5', BEAT], ['f5', BEAT], ['d6', BEAT], ['a5', BEAT], ['a#5',
BEAT * 1.5]]

try:
    speaker.play(liten_mus)

finally: # Turn speaker off if interrupted
    speaker.off()
```

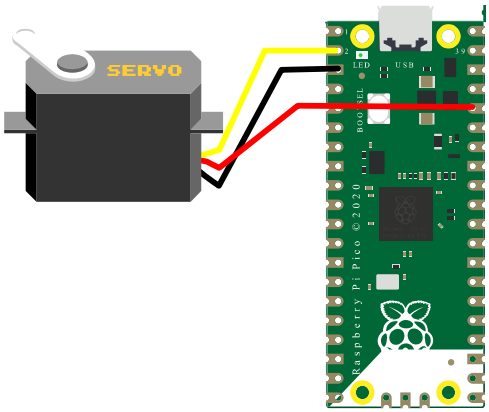
Play individual notes

Play individual notes and control the timing or perform another action:

```
from picozero import Speaker
from time import sleep

speaker = Speaker(5)

BEAT = 0.4

liten_mus = [ ['d5', BEAT / 2], ['d#5', BEAT / 2], ['f5', BEAT], ['d6', BEAT], ['a#5', BEAT],
['d5', BEAT],
              ['f5', BEAT], ['d#5', BEAT], ['d#5', BEAT], ['c5', BEAT / 2],['d5', BEAT / 2],
['d#5', BEAT],
              ['c6', BEAT], ['a5', BEAT], ['d5', BEAT], ['g5', BEAT], ['f5', BEAT], ['f5', BEAT],
['d5', BEAT / 2],
              ['d#5', BEAT / 2], ['f5', BEAT], ['g5', BEAT], ['a5', BEAT], ['a#5', BEAT], ['a5',
BEAT], ['g5', BEAT],
              ['g5', BEAT], ['', BEAT / 2], ['a#5', BEAT / 2], ['c6', BEAT / 2], ['d6', BEAT / 2],
['c6', BEAT / 2],
              ['a#5', BEAT / 2], ['a5', BEAT / 2], ['g5', BEAT / 2], ['a5', BEAT / 2], ['a#5',
BEAT / 2], ['c6', BEAT],
              ['f5', BEAT], ['f5', BEAT], ['f5', BEAT / 2], ['d#5', BEAT / 2], ['d5', BEAT],
['f5', BEAT], ['d6', BEAT],
              ['d6', BEAT / 2], ['c6', BEAT / 2], ['b5', BEAT], ['g5', BEAT], ['g5', BEAT], ['c6',
BEAT / 2],
              ['a#5', BEAT / 2], ['a5', BEAT], ['f5', BEAT], ['d6', BEAT], ['a5', BEAT], ['a#5',
BEAT * 1.5]]

try:
    for note in liten_mus:
        speaker.play(note)
        sleep(0.1) # Leave a gap between notes

finally: # Turn speaker off if interrupted
    speaker.off()
```

Servo

A servo motor connected to a single pin, 3.3v and ground.



Move the servo to its minimum, mid and maximum positions.

```
from picozero import Servo
from time import sleep

servo = Servo(1)

servo.min()
sleep(1)

servo.mid()
sleep(1)

servo.max()
sleep(1)

servo.off()
```

Pulse the servo between its minumum and maximum position.

```
from picozero import Servo

servo = Servo(1)

servo.pulse()
```

Move the servo gradually from its minimum to maximum position in 100 increments.

```
from picozero import Servo
from time import sleep

servo = Servo(1)

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

servo.off()
```

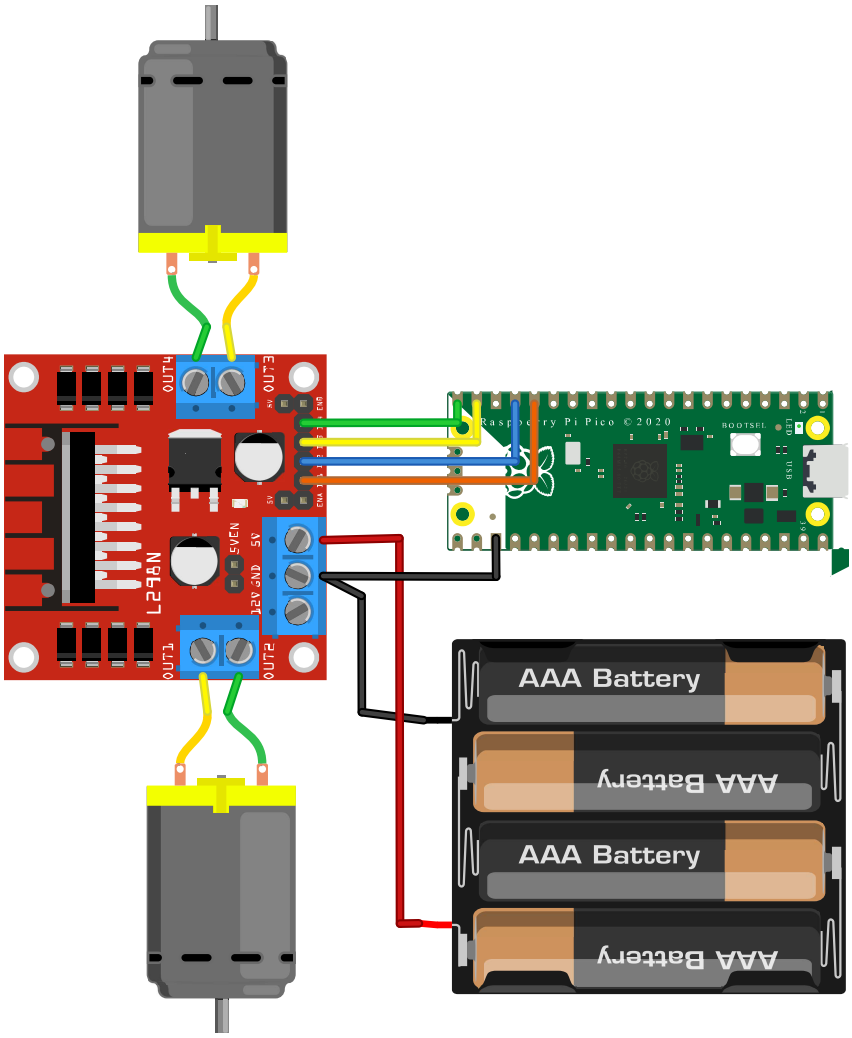
Motor

Move a motor connected via two pins (forward and backward) and a motor controller board:

```
from picozero import Motor
from time import sleep

motor = Motor(14, 15)

motor.move()
sleep(1)
motor.stop()
```

Robot rover

Make a simple two-wheeled robot rover.



Move the rover forward for 1 second and stop:

```
from picozero import Robot
from time import sleep

robot_rover = Robot(left=(14,15), right=(12,13))

# move forward
robot_rover.forward()
sleep(1)
robot_rover.stop()
```

Move the rover (roughly) in a square:

```
from picozero import Robot

robot_rover = Robot(left=(14,15), right=(12,13))

for i in range(4):
    # move forward for 1 second
    robot_rover.forward(t=1, wait=True)
    # rotate to the left for 1 second
    robot_rover.left(t=1, wait=True)
```

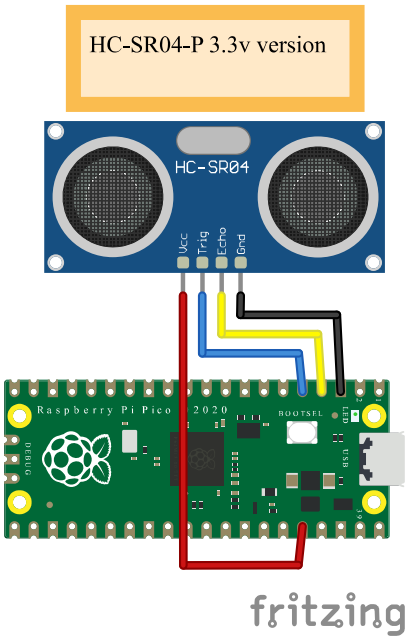
Internal temperature sensor

Check the internal temperature of the Raspberry Pi Pico in degrees Celcius:

```
# Choose View -> Plotter in Thonny to see a graph of the results

from picozero import pico_temp_sensor
from time import sleep

while True:
    print(pico_temp_sensor.temp)
    sleep(0.1)
```

Ultrasonic distance sensor

Get the distance in metres from an ultrasonic distance sensor (HC-SR04):




```
from picozero import DistanceSensor
from time import sleep

ds = DistanceSensor(echo=2, trigger=3)

while True:
    print(ds.distance)
    sleep(0.1)
```