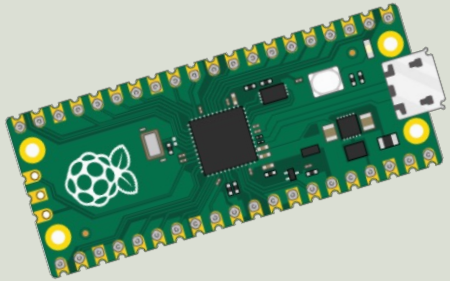
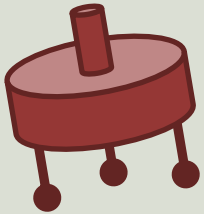
https://www.halvorsen.blog



Raspberry Pi Pico

Potentiometers



Hans-Petter Halvorsen

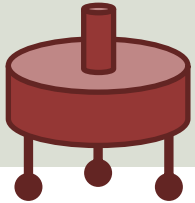
Contents

Introduction

Hans-Petter Halvorsen

Table of Contents

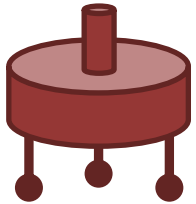
Introduction



- In this Tutorial we will show how we can use a Potentiometer for Raspberry Pi Pico
- We will use MicroPython
- A Potentiometer is basically a variable resistor and Potentiometers change their resistance when you turn a dial/knob
- A Potentiometer has many Applications, we will show some basic examples here

What do you need?

- Raspberry Pi Pico
- A Micro-USB cable
- A PC with Thonny Python Editor (or another Python Editor)
- Breadboard
- Electronics Components like LED, Resistors, Jumper wires, etc.
- Potentiometer(s)



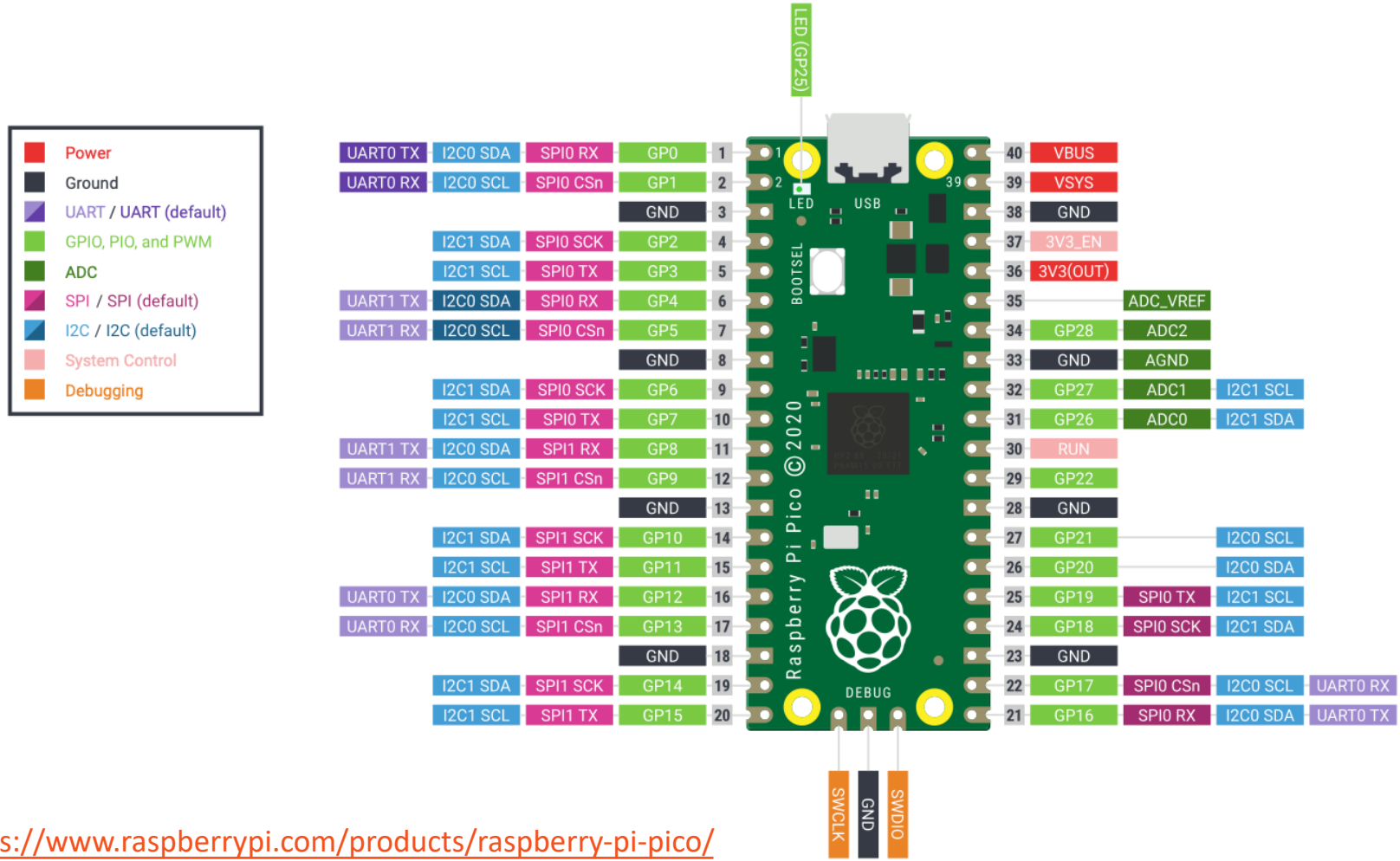
Raspberry Pi Pico

- Raspberry Pi Pico is a microcontroller board developed by the Raspberry Pi Foundation
- Raspberry Pi Pico has similar features as Arduino devices
- Raspberry Pi Pico is typically used for Electronics projects, IoT Applications, etc.
- You typically use MicroPython, which is a downscaled version of Python, in order to program it

https://www.raspberrypi.com/products/raspberry-pi-pico/

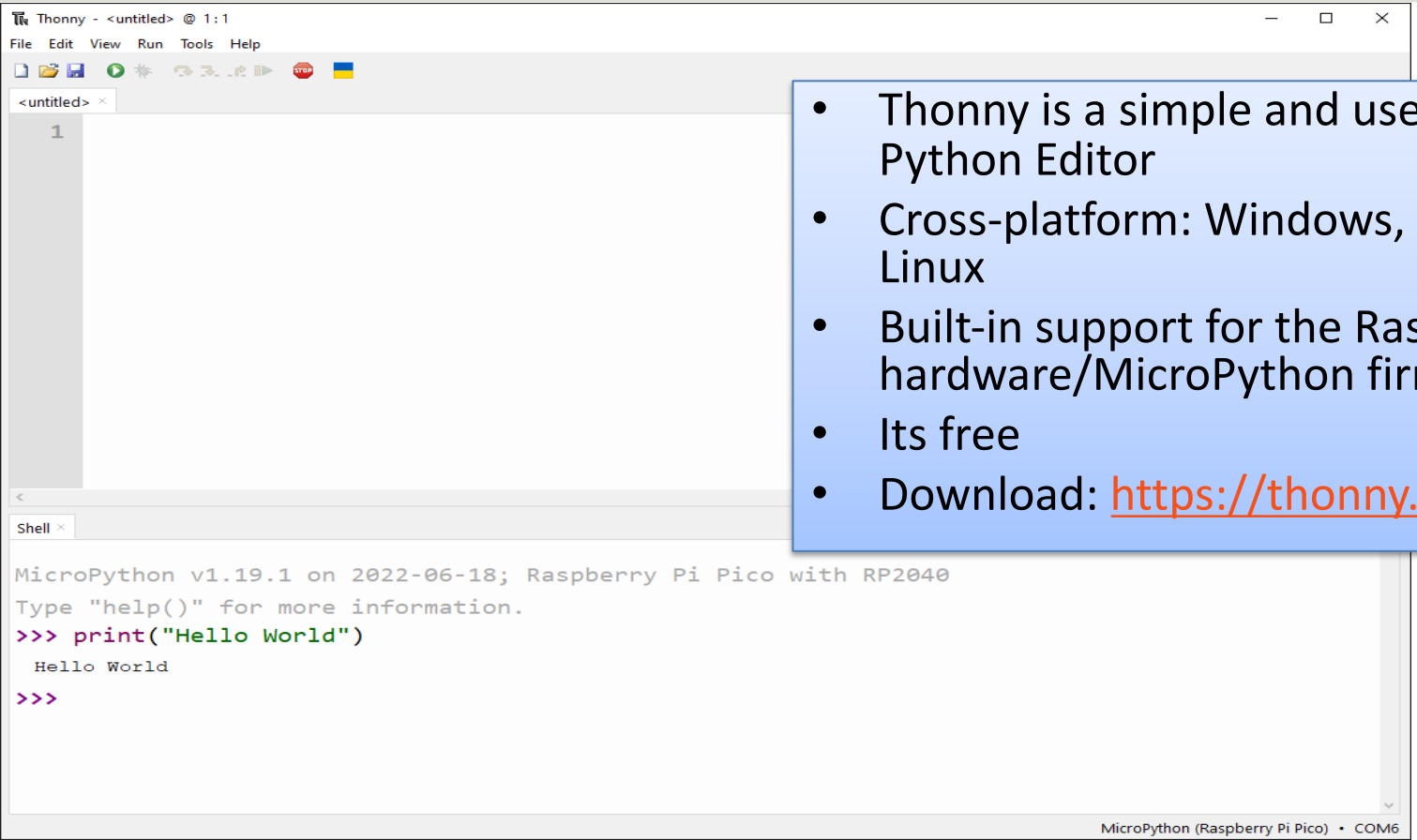
https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico

Pico Pinout



https://www.raspberrypi.com/products/raspberry-pi-pico/

Thonny



- Thonny is a simple and user-friendly Python Editor
- Cross-platform: Windows, macOS and Linux
- Built-in support for the Raspberry Pi Pico hardware/MicroPython firmware
- Its free
- Download: https://thonny.org

MicroPython

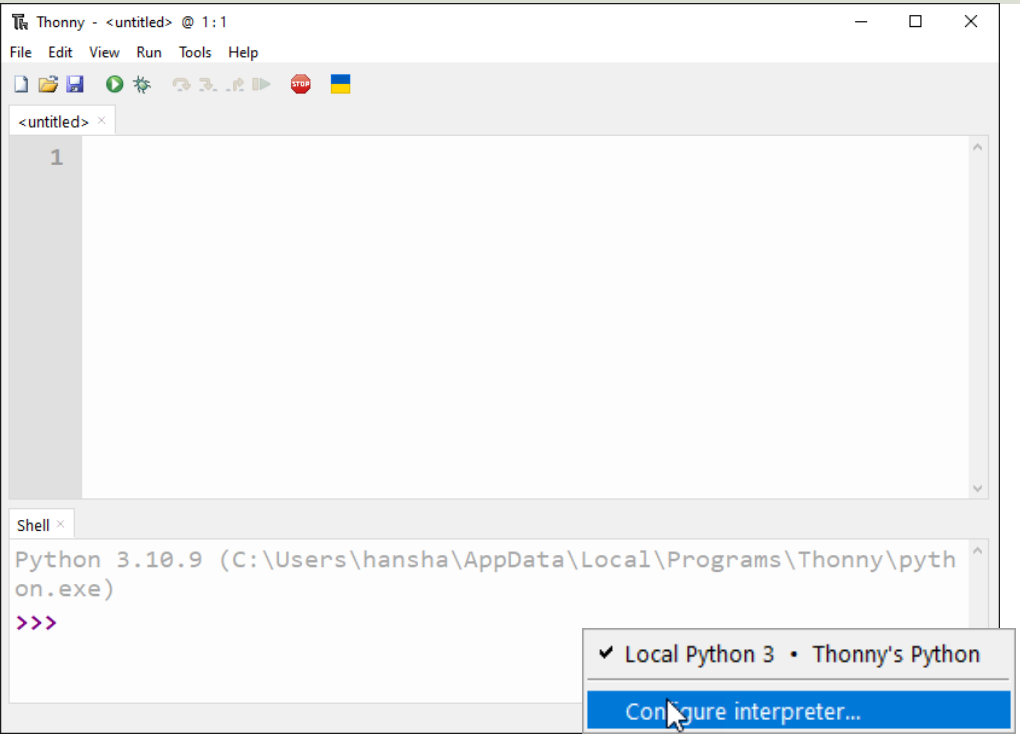
- MicroPython is a downscaled version of Python
- It is typically used for Microcontrollers and constrained systems (low memory, etc.)
- Examples of such Microcontrollers that have tailormade MicroPython firmwares are Raspberry Pi Pico and Micro:bit

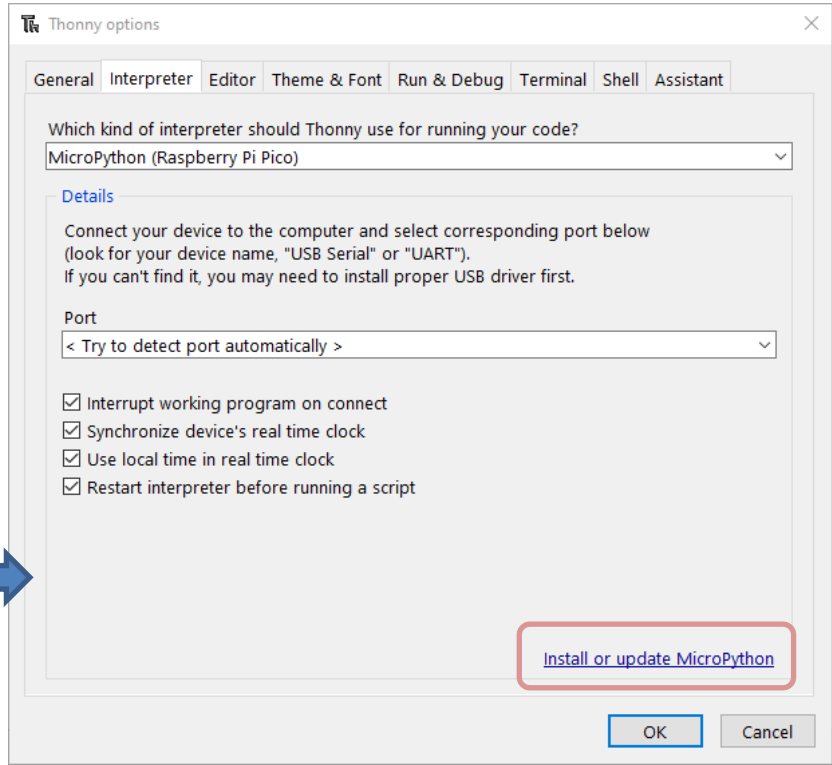
https://docs.micropython.org/en/latest/index.html https://micropython.org

MicroPython Firmware

- The first time you need to install the MicroPython Firmware on your Raspberry Pi Pico
- You can install the MicroPython Firmware manually or you can use the Thonny Editor

Install MicroPython Firmware using Thonny





https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/3

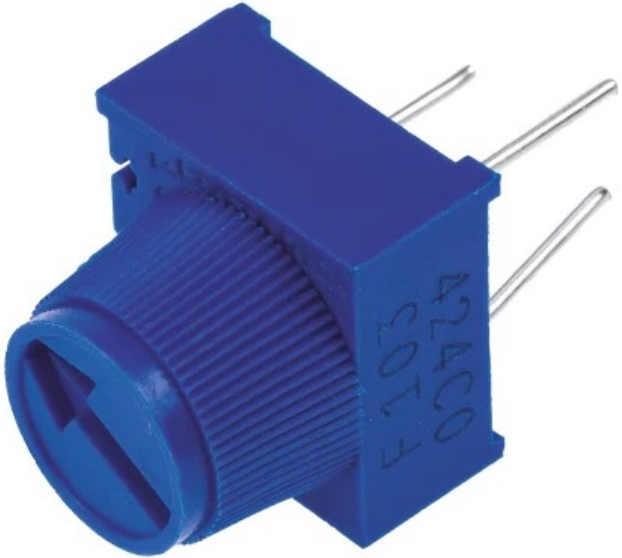
Potentiometer

Hans-Petter Halvorsen

Table of Contents

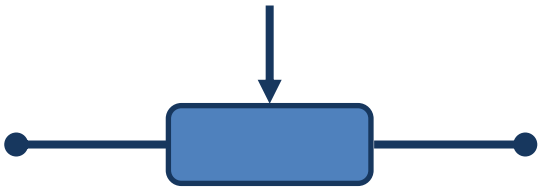
Potentiometer

Potentiometer Examples:

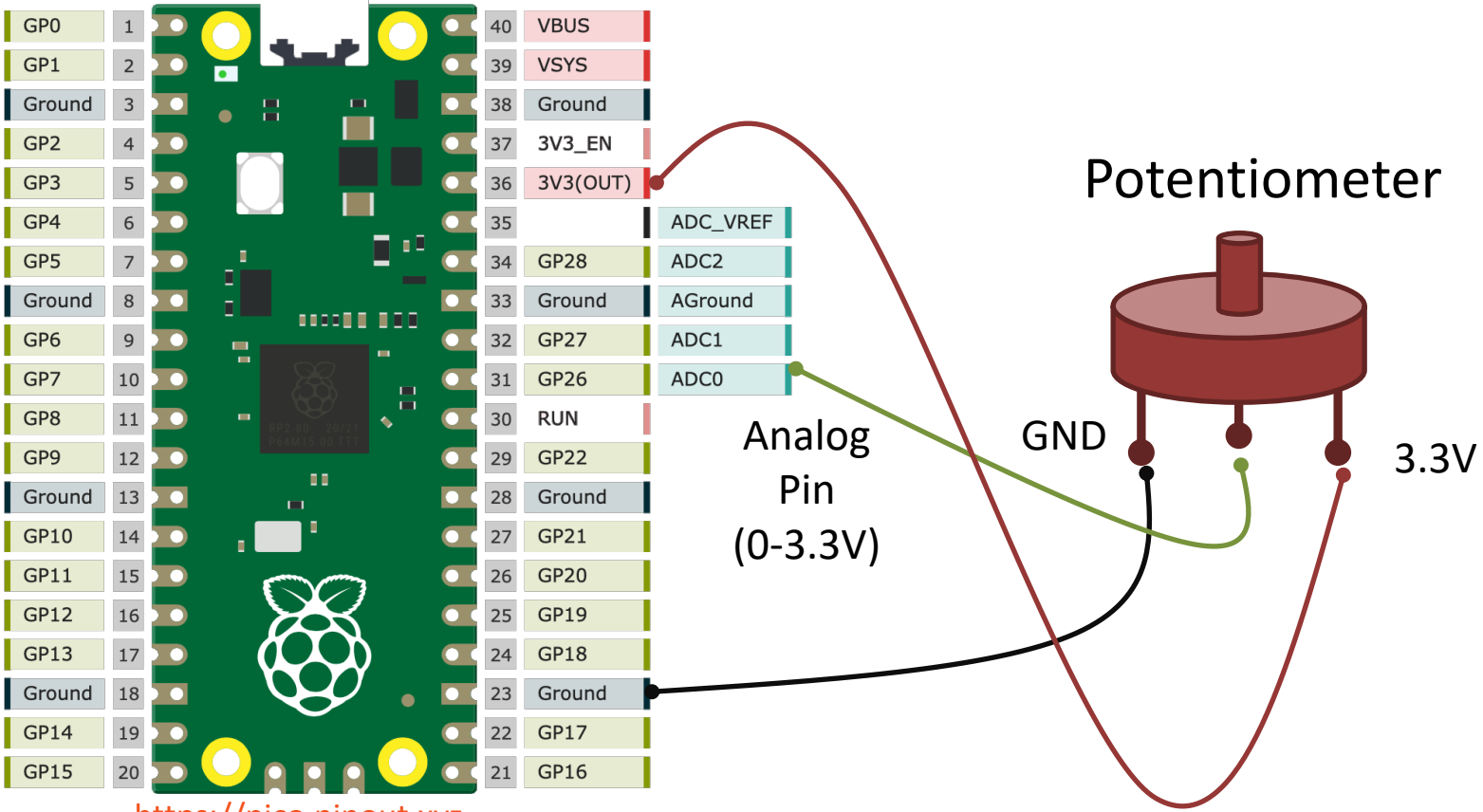


A Potentiometer is basically a **variable resistor** and Potentiometers change their resistance when you turn a dial/knob. A Potentiometer has 3 legs.

Potentiometer Symbol:

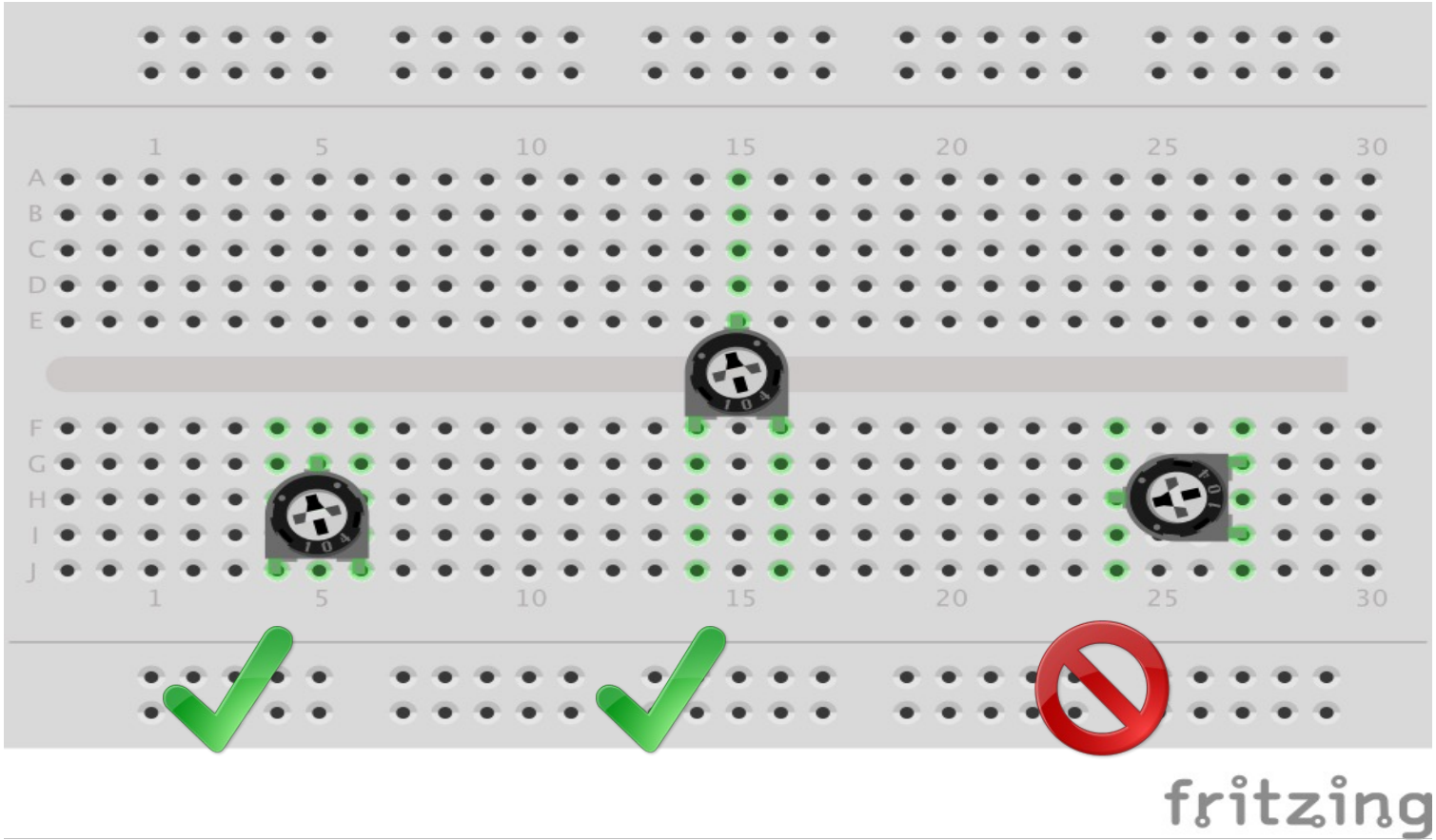


Potentiometer Wiring



https://pico.pinout.xyz

Correct use of Potentiometer on Breadboard



https://www.halvorsen.blog

MicroPython Code Examples

Potentiometer

Hans-Petter Halvorsen

Table of Contents

Potentiometer

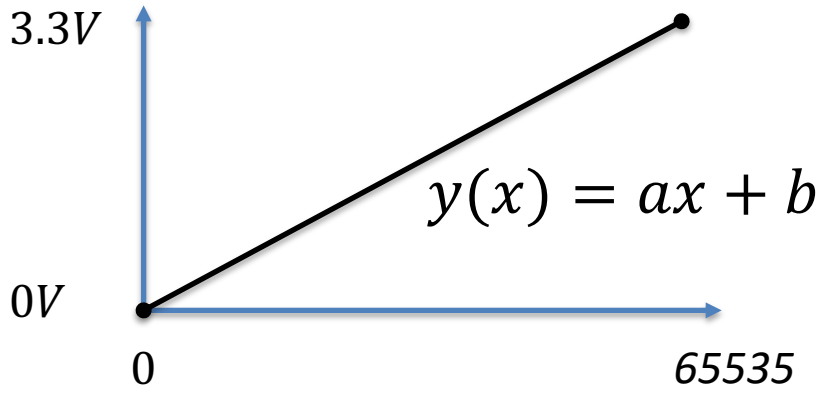
```
from machine import ADC
from time import sleep

adcpin = 26
pot = ADC(adcpin)

while True:
    adc_value = pot.read_u16()
    print(adc_value)

    volt = (3.3/65535)*adc_value
    print(round(volt,2))

    sleep(1)
```

ADC Value to Voltage Value

Analog Pins: The built-in Analog-to-Digital Converter (ADC) on Pico is 16bit, producing values from 0 to 65535.

The `read_u16()` function gives a value between 0 and 65535. It must be converted to a Voltage Signal 0 - 3.3v

ADC = 0 -> 0v
ADC = 65535 -> 3.3v



This gives the following conversion formula:

$$y(x) = \frac{3.3}{65535}x$$

Code v2

```
from machine import ADC
from time import sleep

def ReadPotentiometer():
    adcpin = 26
    pot = ADC(adcpin)

    adc_value = pot.read_u16()
    volt = (3.3/65535)*adc_value

    percentPot = ScalePercent(volt)

    return percentPot

def ScalePercent(volt):
    percent = (volt/3.3)*100
    return int(percent)

while True:
    potvalue = ReadPotentiometer()
    print(potvalue)
    sleep(1)
```

This code reads values from the Potentiometer and converts it to a value between 0 and 100%.
2 functions have been made to make the code more structured and reusable

Thonny - C:\Users\hansha\OneDrive\Documents\Industrial IT and Automation\IoT\Raspberry Pi Pico\Code Examples\potentiometer2.py @ 17 : 29

File Edit View Run Tools Help

potentiometer.py potentiometer2.py

```
from machine import ADC
from time import sleep

def ReadPotentiometer():
    adcpin = 26
    pot = ADC(adcpin)

    adc_value = pot.read_u16()
    volt = (3.3/65535)*adc_value

    percentPot = ScalePercent(volt)

    return percentPot

def ScalePercent(volt):
    #0-3.3V->0-100%
    percent = (volt/3.3)*100
    return int(percent)

while True:
    potvalue = ReadPotentiometer()
    print(potvalue)
    sleep(1)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
  0
  10
  19
  35
  55
  65
  79
  100
```

MicroPython (Raspberry Pi Pico) • COM6

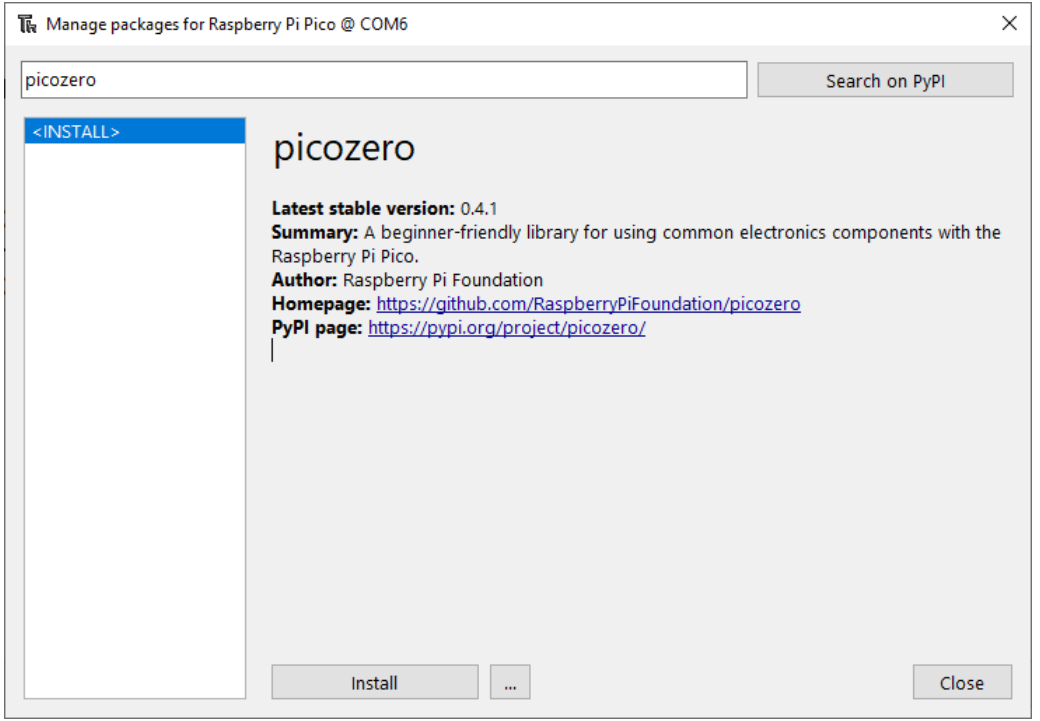
PicoZero

Hans-Petter Halvorsen

Table of Contents

PicoZero

- The **picozero** Python Library is intended to be a beginner-friendly library for using common electronics components with the Raspberry Pi Pico
- It can be used instead of the machine Library in many cases
- You install it like an ordinary Python Library using “pip install picozero” or from the “Manage Packages” window in the Thonny editor



https://pypi.org/project/picozero/
https://picozero.readthedocs.io
https://github.com/RaspberryPiFoundation/picozero

Picozero + Potentiometer

https://picozero.readthedocs.io/en/latest/api.html#potentiometer-pot

PicoZero Potentiometer

```
from picozero import Pot
from time import sleep

adcpin = 26
pot = Pot(adcpin)

while True:
    potvalue = pot.value
    print(round(potvalue,2))

    potvoltage = pot.voltage
    print(round(potvoltage,2))

    sleep(0.5)
```

Value between 0 (Min) and 1 (Max)

Value between 0 (Min) and 3.3 (Max)

Potentiometer and LED

Hans-Petter Halvorsen

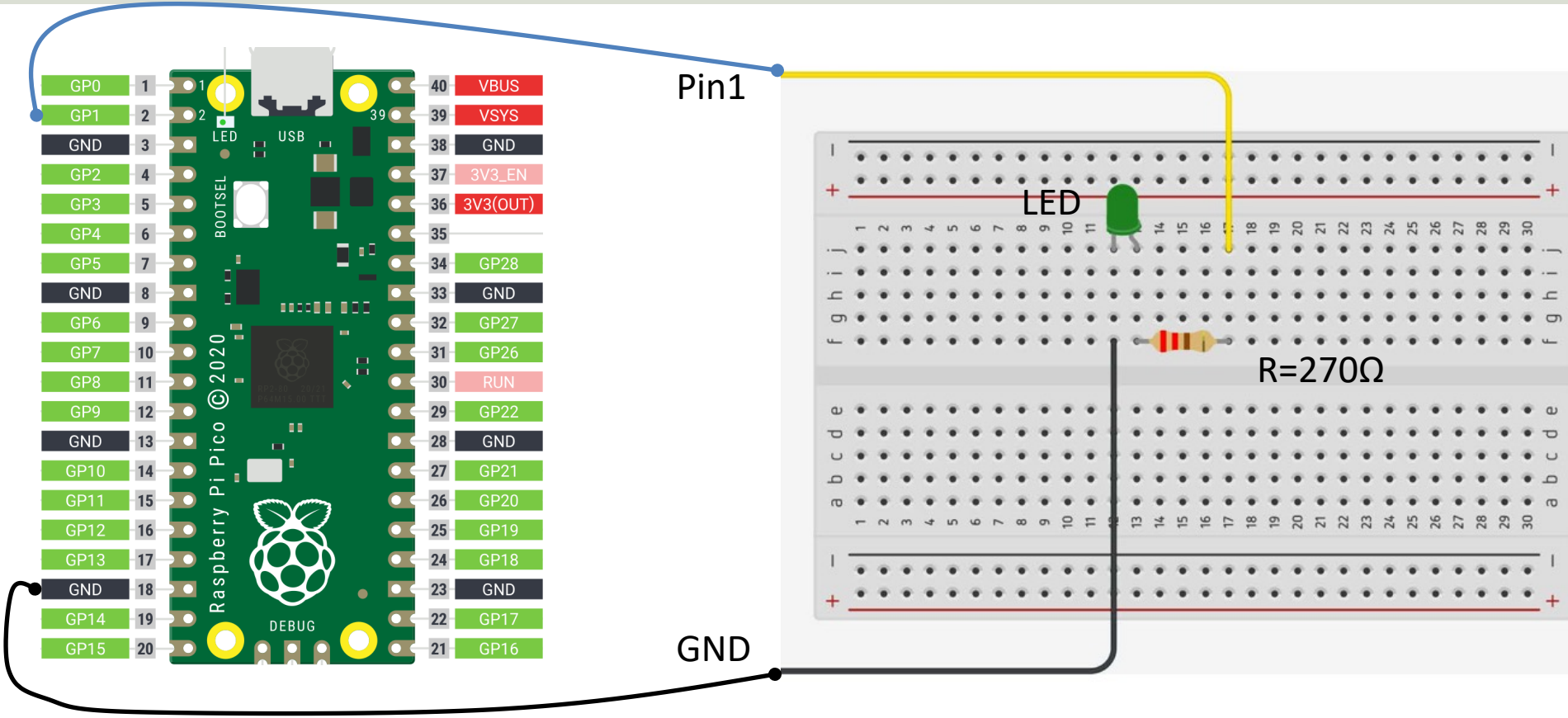
Table of Contents

LED Examples

Here we will show some examples where we combine a Potentiometer and a LED

- Use the Potentiometer to control the Brightness of the LED using Pulse Width Modulation (PWM)
- Use the Potentiometer to control how fast the LED should blink

Wiring the LED



LED Brightness Example

- We use the Potentiometer to control the Brightness of the LED using Pulse Width Modulation (PWM)

LED Brightness

```
from machine import ADC, Pin, PWM
from time import sleep

adcpin = 26
pot = ADC(adcpin)

ledpin = 1
pwm = PWM(Pin(ledpin))
pwm.freq(1000)

while True:
    adc_value = pot.read_u16()
    pwm.duty_u16(adc_value)
    sleep(0.1)
```

LED Blinking Speed Example

- We use the Potentiometer to control how fast the LED should blink
- We have made a Potentiometer function that gives a value between 0 and 100%
- Then we have made a Speed function that says
  - If 0% -> Wait 5s (Slowest LED Speed)
  - If 100%-> Wait 0.5s (Fastest LED Speed)

$$y = -\frac{4.5}{100}x + 5$$

We have used the following formula: $$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

LED Blinking Speed

```
from machine import ADC, Pin
from time import sleep

ledpin = 1
led = Pin(ledpin, Pin.OUT)

def ReadPotentiometer():
    adcpin = 26
    pot = ADC(adcpin)

    adc_value = pot.read_u16()
    volt = (3.3/65535)*adc_value
    percentPot = ScalePercent(volt)
    return percentPot

def ScalePercent(volt):
    percent = (volt/3.3)*100
    return percent

def BlinkSpeed(x):
    y = -(4.5/100)*x + 5
    y = round(y, 1)
    return y

while True:
    led.toggle()
    potvalue = ReadPotentiometer()
    waitTime = BlinkSpeed(potvalue)
    sleep(waitTime)
```

Raspberry Pi Pico Resources

- Raspberry Pi Pico:
  https://www.raspberrypi.com/products/raspberry-pi-pico/
- Raspberry Pi Foundation:
  https://projects.raspberrypi.org/en/projects?hardware[]=pico
- Getting Started with Pico:
  https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico
- MicroPython:
  https://docs.micropython.org/en/latest/index.html

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog