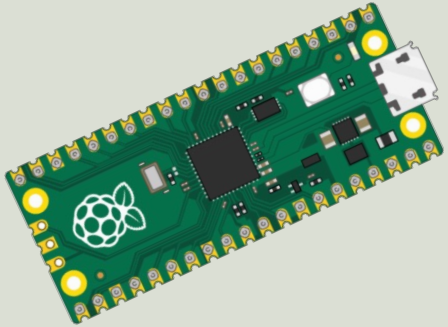
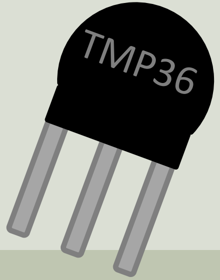
https://www.halvorsen.blog



Raspberry Pi Pico

TMP36 Temperature Sensor



Hans-Petter Halvorsen

Contents

- Introduction
- Raspberry Pi Pico
  - Thonny Python Editor
  - MicroPython
- TMP36 Temperature Sensor

Introduction

Introduction

- In this Tutorial we will read values from a **TMP36** Temperature Sensor.
- We will use a **Raspberry Pi Pico** and **MicroPython.**
- We will use the **Thonny** Python Editor which has built-in support for the Raspberry Pi Pico hardware/MicroPython firmware.

What do you need?

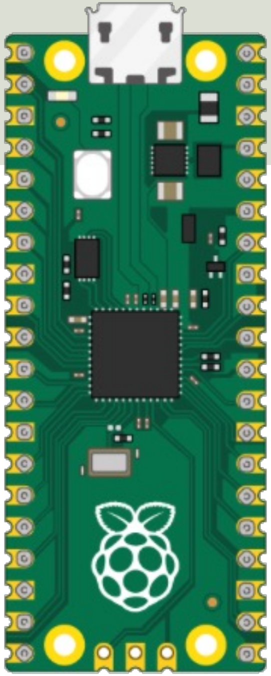
- Raspberry Pi Pico
- A Micro-USB cable
- A PC with Thonny Python Editor (or another Python Editor)
- Breadboard
- Electronics Components like LED, Resistors, Jumper wires, etc.
- TMP36 Temperature Sensor

Raspberry Pi Pico

Hans-Petter Halvorsen

Table of Contents

Raspberry Pi Pico



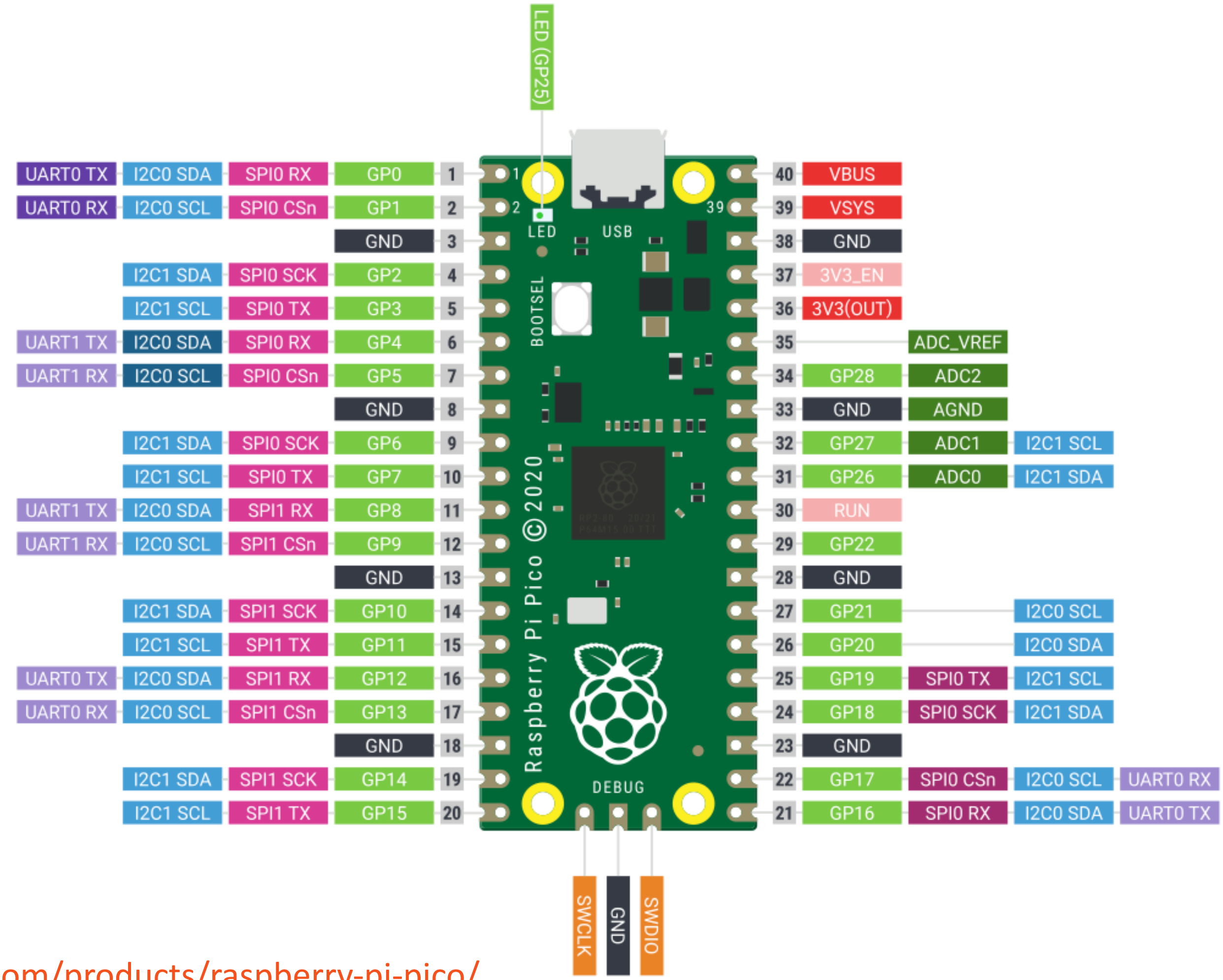
- Raspberry Pi Pico is a microcontroller board developed by the Raspberry Pi Foundation
- Raspberry Pi Pico has similar features as Arduino devices
- Raspberry Pi Pico is typically used for Electronics projects, IoT Applications, etc.
- You typically use MicroPython, which is a downscaled version of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

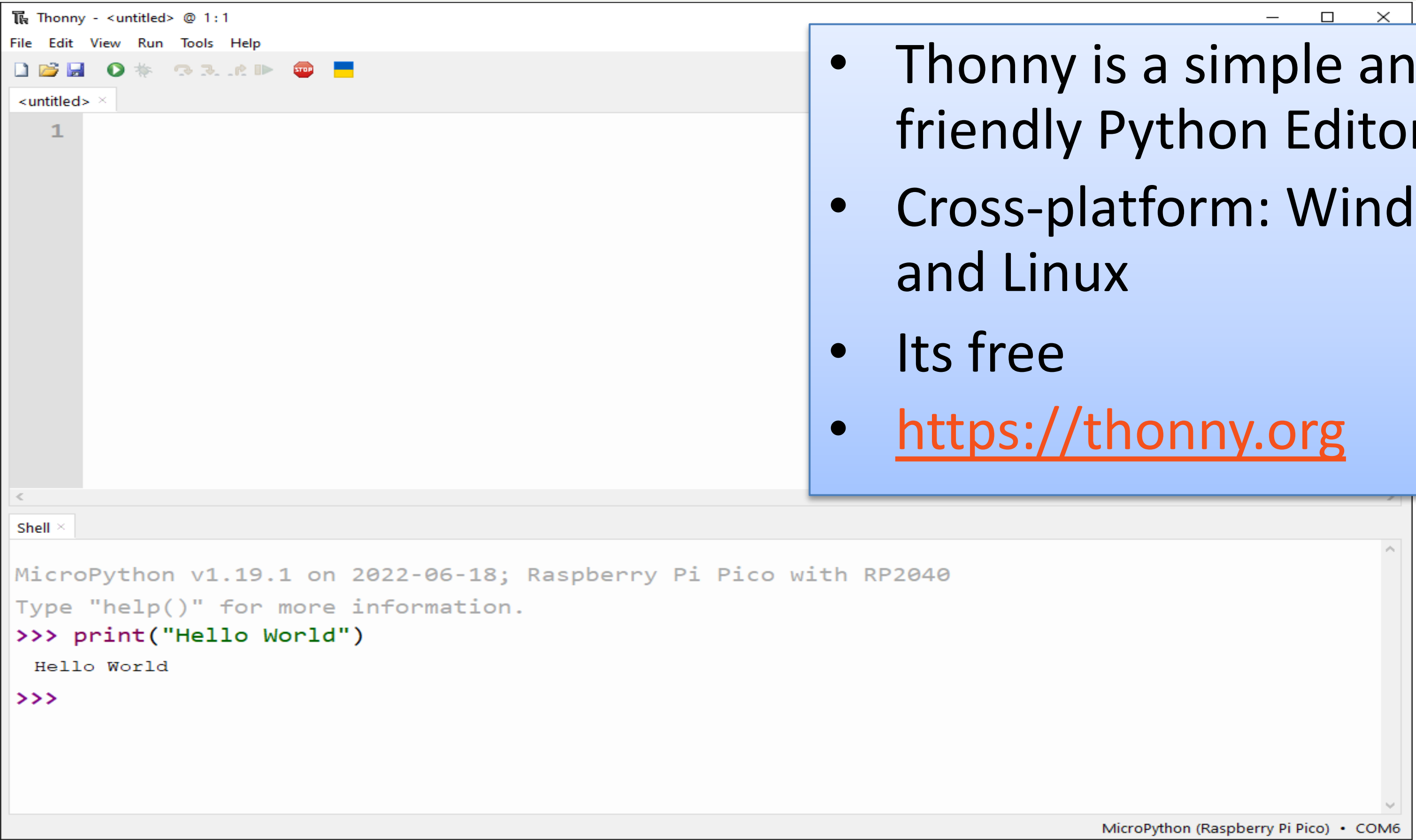
- 🟥 Power
- ⬛ Ground
- ▨ UART / UART (default)
- 🟩 GPIO, PIO, and PWM
- 🟩 ADC
- ▨ SPI / SPI (default)
- ▨ I2C / I2C (default)
- 🟥 System Control
- 🟧 Debugging



UART0 TX	I2C0 SDA	SPI0 RX	GP0	1	40	VBUS			
UART0 RX	I2C0 SCL	SPI0 CSn	GP1	2	39	VSYS			
			GND	3	38	GND			
	I2C1 SDA	SPI0 SCK	GP2	4	37	3V3_EN			
	I2C1 SCL	SPI0 TX	GP3	5	36	3V3(OUT)			
UART1 TX	I2C0 SDA	SPI0 RX	GP4	6	35		ADC_VREF		
UART1 RX	I2C0 SCL	SPI0 CSn	GP5	7	34	GP28	ADC2		
			GND	8	33	GND	AGND		
	I2C1 SDA	SPI0 SCK	GP6	9	32	GP27	ADC1	I2C1 SCL	
	I2C1 SCL	SPI0 TX	GP7	10	31	GP26	ADC0	I2C1 SDA	
UART1 TX	I2C0 SDA	SPI1 RX	GP8	11	30	RUN			
UART1 RX	I2C0 SCL	SPI1 CSn	GP9	12	29	GP22			
			GND	13	28	GND			
	I2C1 SDA	SPI1 SCK	GP10	14	27	GP21		I2C0 SCL	
	I2C1 SCL	SPI1 TX	GP11	15	26	GP20		I2C0 SDA	
UART0 TX	I2C0 SDA	SPI1 RX	GP12	16	25	GP19	SPI0 TX	I2C1 SCL	
UART0 RX	I2C0 SCL	SPI1 CSn	GP13	17	24	GP18	SPI0 SCK	I2C1 SDA	
			GND	18	23	GND			
	I2C1 SDA	SPI1 SCK	GP14	19	22	GP17	SPI0 CSn	I2C0 SCL	UART0 RX
	I2C1 SCL	SPI1 TX	GP15	20	21	GP16	SPI0 RX	I2C0 SDA	UART0 TX

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

Thonny


Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
<untitled>
Shell
MicroPython (Raspberry Pi Pico) • COM6

```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> print("Hello World")
 Hello World
>>>
```

- Thonny is a simple and user-friendly Python Editor
- Cross-platform: Windows, macOS and Linux
- Its free
- https://thonny.org

MicroPython

- MicroPython is a downscaled version of Python
- It is typically used for Microcontrollers and constrained systems

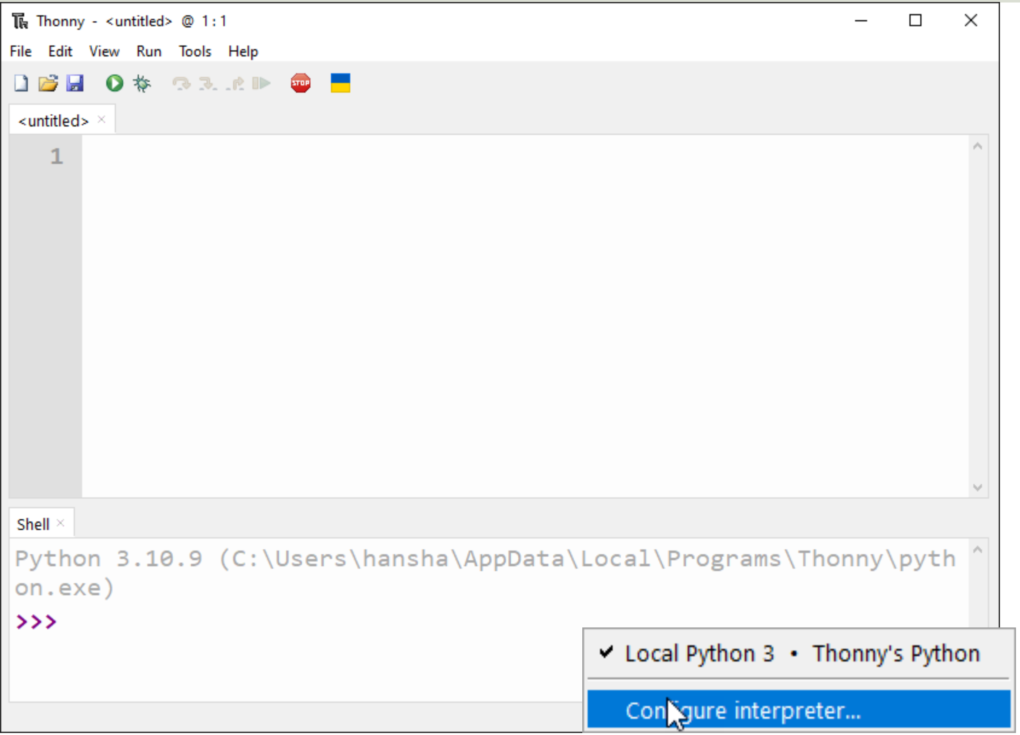
https://docs.micropython.org/en/latest/index.html

https://micropython.org

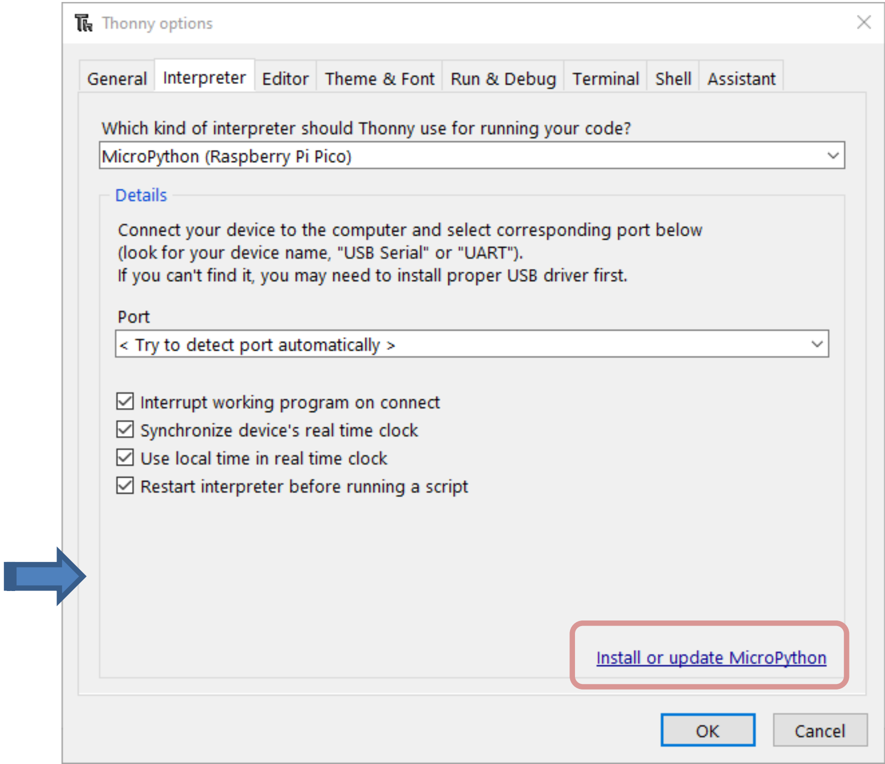
MicroPython Firmware

- 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



```
Python 3.10.9 (C:\Users\hansha\AppData\Local\Programs\Thonny\python.exe)
>>>
```



Which kind of interpreter should Thonny use for running your code?

MicroPython (Raspberry Pi Pico)

Connect your device to the computer and select corresponding port below
(look for your device name, "USB Serial" or "UART").
If you can't find it, you may need to install proper USB driver first.

Port

< Try to detect port automatically >

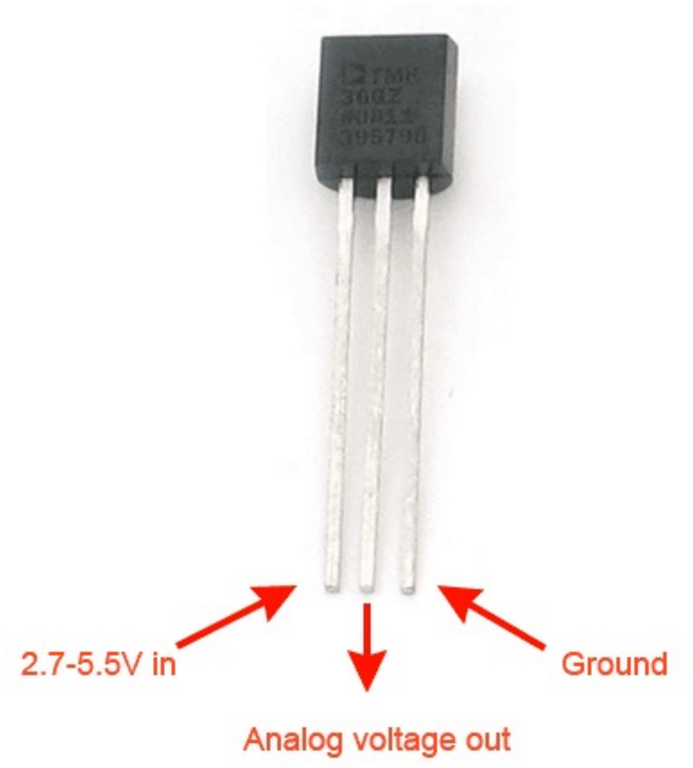
☑ Interrupt working program on connect
☑ Synchronize device's real time clock
☑ Use local time in real time clock
☑ Restart interpreter before running a script

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

TMP36 Temperature Sensor

Table of Contents

TMP36 Temperature Sensor

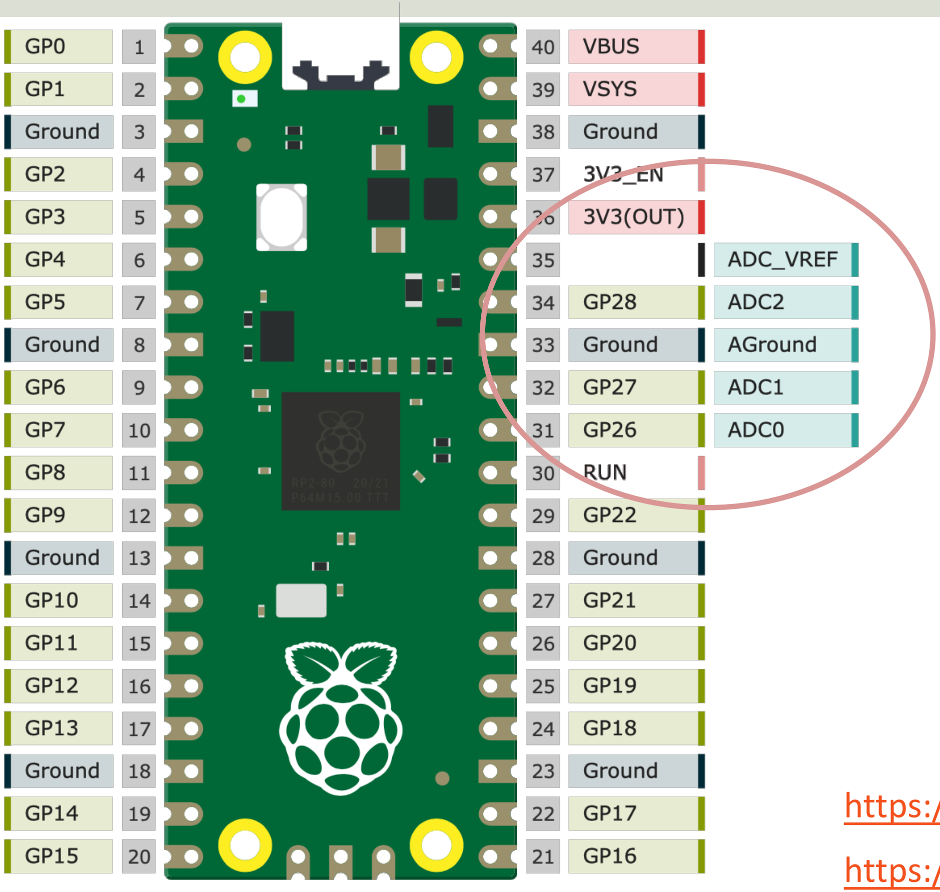


A Temperature sensor like TM36 use a solid-state technique to determine the temperature.

They use the fact as temperature increases, the voltage across a diode increases at a known rate.

https://learn.adafruit.com/tmp36-temperature-sensor

Analog Values with Pico



Raspberry Pi Pico has 3 Analog Inputs (ADC)

ADC 0 – Pin 26

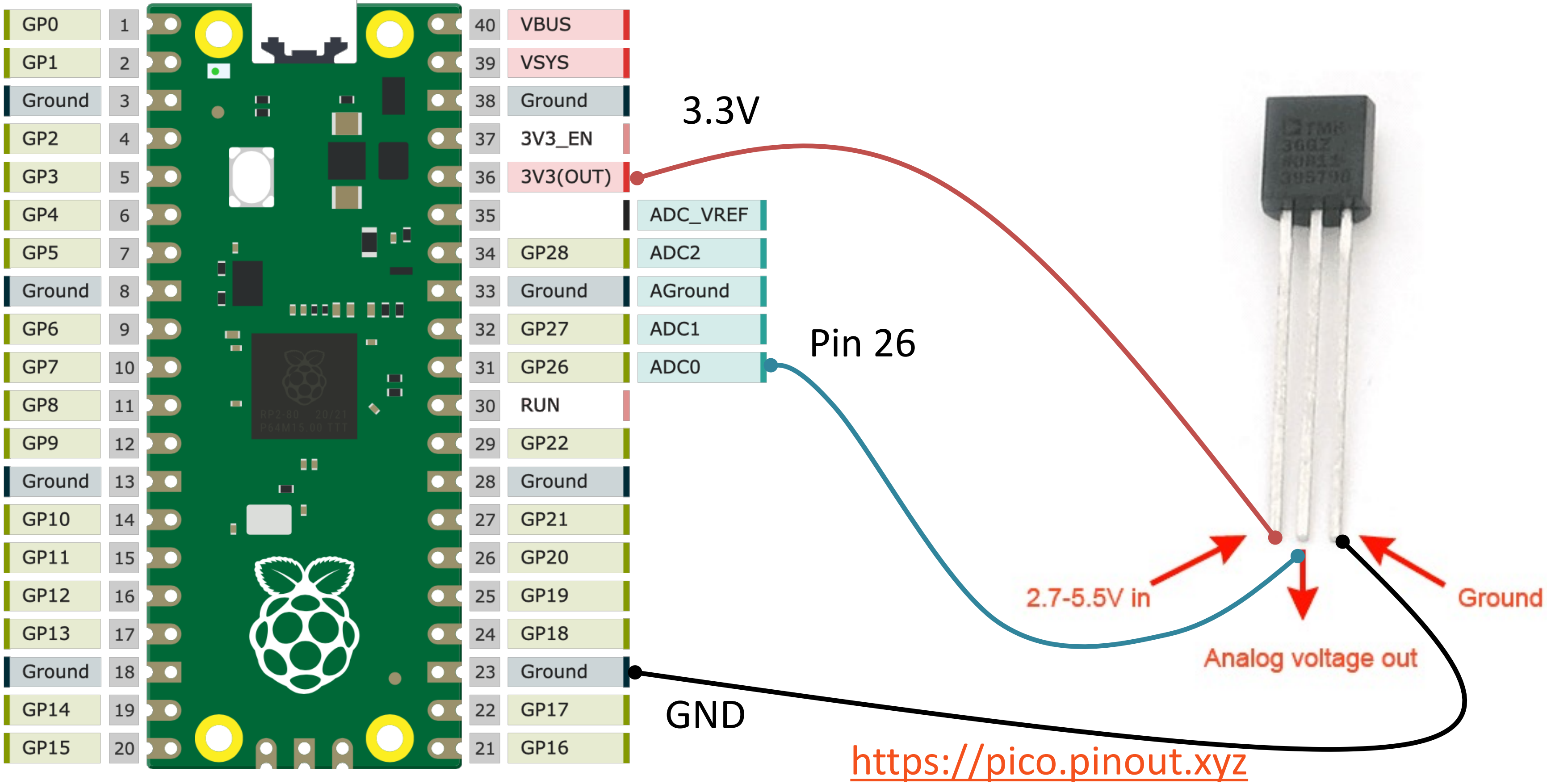
ADC 1 – Pin 27

ADC 2 – Pin 28

https://pico.pinout.xyz

https://docs.micropython.org/en/latest/library/machine.ADC.html

TMP36 Wiring



GP0	1	40	VBUS	
GP1	2	39	VSYS	
Ground	3	38	Ground	
GP2	4	37	3V3_EN	
GP3	5	36	3V3(OUT)	
GP4	6	35		ADC_VREF
GP5	7	34	GP28	ADC2
Ground	8	33	Ground	AGround
GP6	9	32	GP27	ADC1
GP7	10	31	GP26	ADC0
GP8	11	30	RUN	
GP9	12	29	GP22	
Ground	13	28	Ground	
GP10	14	27	GP21	
GP11	15	26	GP20	
GP12	16	25	GP19	
GP13	17	24	GP18	
Ground	18	23	Ground	
GP14	19	22	GP17	
GP15	20	21	GP16	

https://pico.pinout.xyz

Main Code Structure

1. Initialization.

2. Read from ADC (Analog to Digital Converter) using “read_u16()” function.

3. Convert raw ADC Value (0-65535) to Voltage Value (0-3.3v). The built-in ADC has 16 resolution.

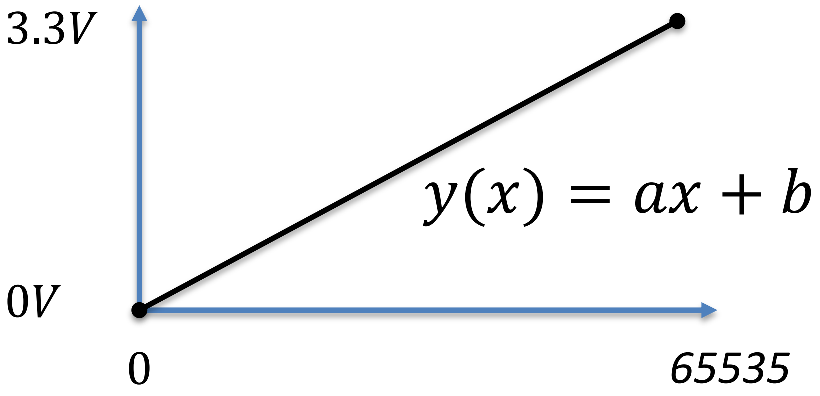
4. Convert from Voltage Value to Temperature in degrees Celsius. Use information from the TMP36 Temperature Sensor Datasheet.

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

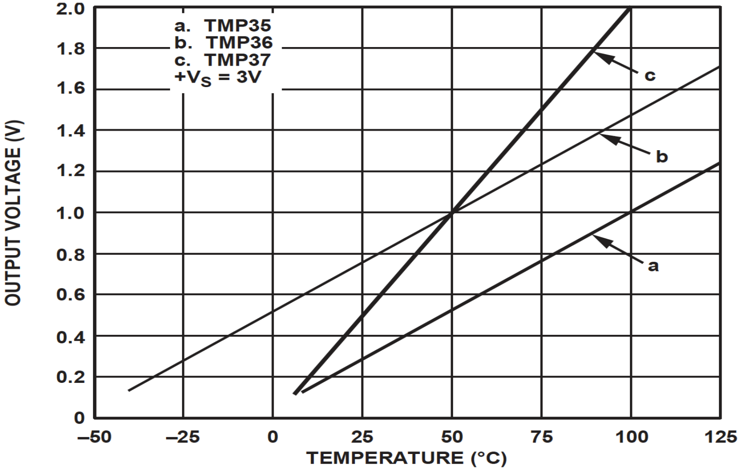
ADC = 0 -> 0v
ADC = 65535 -> 3.3v



This gives the following conversion formula:

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

Voltage to degrees Celsius



Convert form Voltage (V) to degrees Celsius
From the **Datasheet** we have:

$$(x_1, y_1) = (0.75V, 25°C)$$
$$(x_2, y_2) = (1V, 50°C)$$

There is a linear relationship between Voltage and degrees Celsius:

$$y = ax + b$$

We can find a and b using the following known formula:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

This gives:

$$y - 25 = \frac{50 - 25}{1 - 0.75}(x - 0.75)$$

Then we get the following formula:

$$y = 100x - 50$$

Datasheet: https://cdn-learn.adafruit.com/assets/assets/000/010/131/original/TMP35_36_37.pdf

Main Code Structure

1. Initialization

```
adcpin = 26
tmp36 = ADC(adcpin)
```

2. Read from ADC:

```
adc_value = tmp36.read_u16()
```

3. Convert raw ADC Value (0-65535) to Voltage Value (0-3.3v):

```
volt = (3.3/65535)*adc_value
```

4. Convert from Voltage Value to Temperature in degrees Celsius:

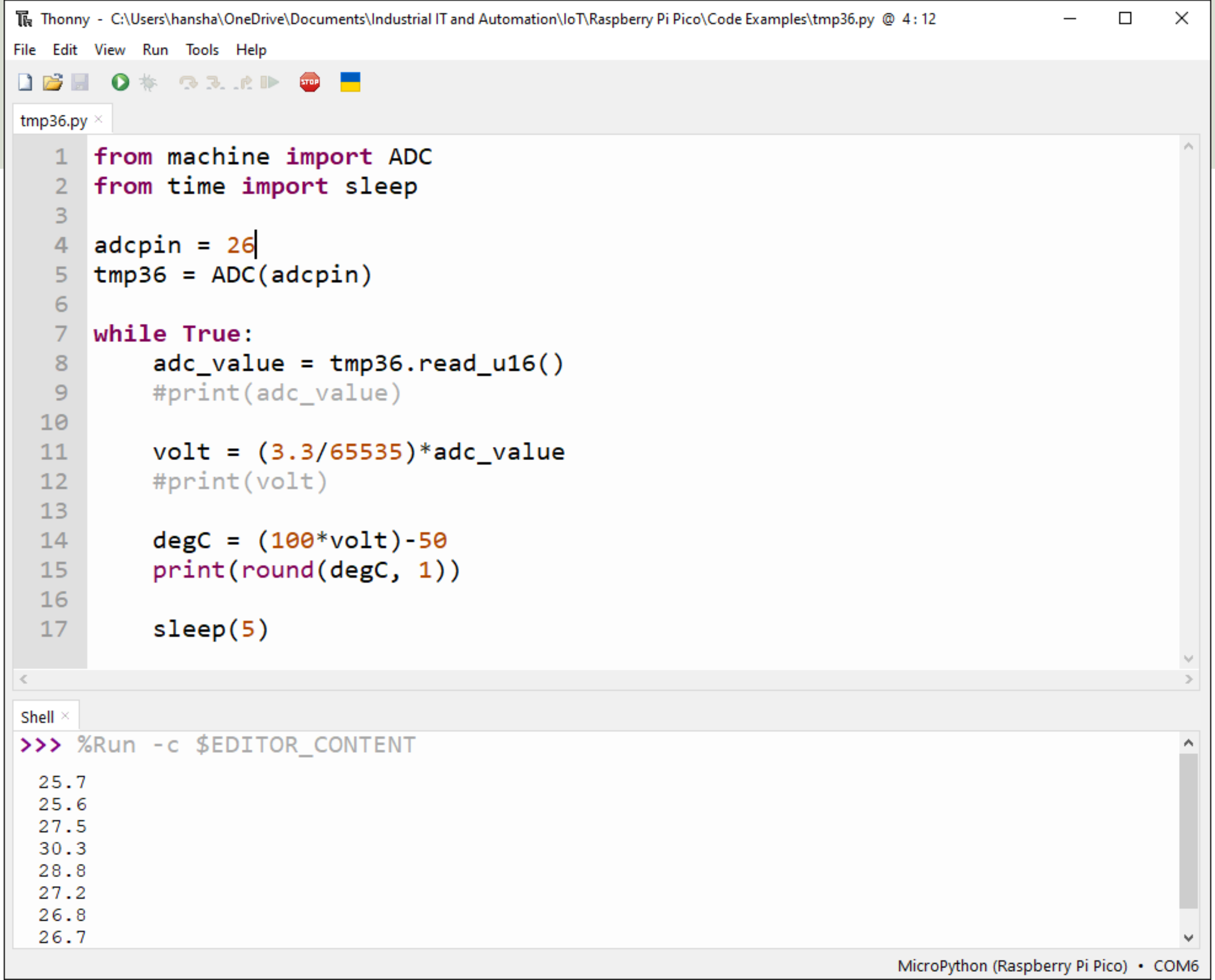
```
degC = (100*volt)-50
```

TMP36 Example

```
from machine import ADC
from time import sleep

adcpin = 26
tmp36 = ADC(adcpin)

while True:
    adc_value = tmp36.read_u16()
    volt = (3.3/65535)*adc_value
    degC = (100*volt)-50
    print(round(degC, 1))
    sleep(5)
```


Thonny - C:\Users\hansha\OneDrive\Documents\Industrial IT and Automation\IoT\Raspberry Pi Pico\Code Examples\tmp36.py @ 4 : 12
File
Edit
View
Run
Tools
Help
tmp36.py
Shell
MicroPython (Raspberry Pi Pico) • COM6

```python
from machine import ADC
from time import sleep

adcpin = 26
tmp36 = ADC(adcpin)

while True:
    adc_value = tmp36.read_u16()
    #print(adc_value)

    volt = (3.3/65535)*adc_value
    #print(volt)

    degC = (100*volt)-50
    print(round(degC, 1))

    sleep(5)
```

```
>>> %Run -c $EDITOR_CONTENT

 25.7
 25.6
 27.5
 30.3
 28.8
 27.2
 26.8
 26.7
```

TMP36 Code v2

```
from machine import ADC
from time import sleep

adcpin = 26
tmp36 = ADC(adcpin)

def ReadTemperature():
    adc_value = tmp36.read_u16()
    volt = (3.3/65535)*adc_value
    degC = (100*volt)-50
    return degC

while True:
    degC = ReadTemperature()
    print(round(degC, 1))
    sleep(5)
```

TMP36 Code v3

TemperatureSensors.py

```
from machine import ADC

class Tmp36Sensor:
    def __init__(self, pin):
        self.tmp36 = ADC(pin)

    def ReadTemperature(self):
        adc_value = self.tmp36.read_u16()
        volt = (3.3/65535)*adc_value
        degC = (100*volt)-50
        return round(degC, 1)
```

```
from TemperatureSensors import Tmp36Sensor
from time import sleep

adcpin = 26
tmp36 = Tmp36Sensor(adcpin)

while True:
    degC = tmp36.ReadTemperature()
    print(degC)
    sleep(5)
```

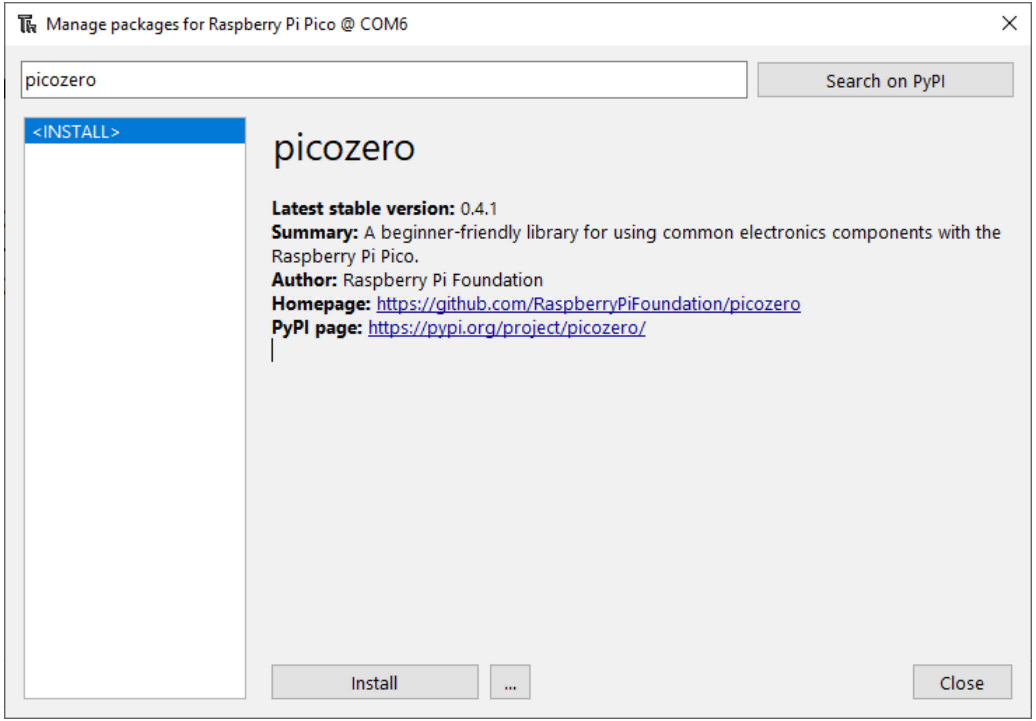
PicoZero

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

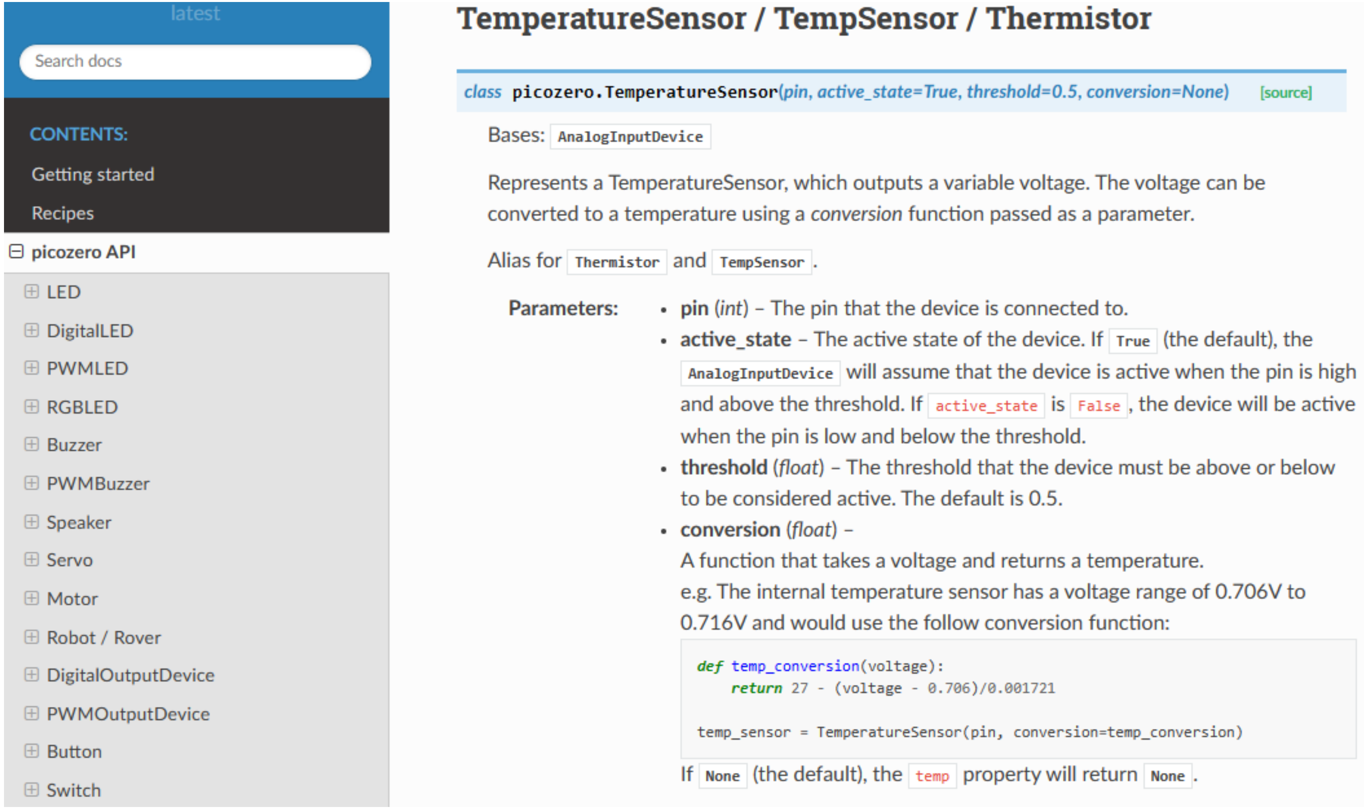


picozero

Latest stable version: 0.4.1
Summary: A beginner-friendly library for using common electronics components with the Raspberry Pi Pico.
Author: Raspberry Pi Foundation
Homepage: https://github.com/RaspberryPiFoundation/picozero
PyPI page: https://pypi.org/project/picozero/

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

PicoZero TemperatureSensor



TemperatureSensor / TempSensor / Thermistor

class `picozero.TemperatureSensor`(*pin, active_state=True, threshold=0.5, conversion=None*) [source]

Bases: `AnalogInputDevice`

Represents a TemperatureSensor, which outputs a variable voltage. The voltage can be converted to a temperature using a *conversion* function passed as a parameter.

Alias for `Thermistor` and `TempSensor`.

Parameters:

- **pin** (*int*) – The pin that the device is connected to.
- **active_state** – The active state of the device. If `True` (the default), the `AnalogInputDevice` will assume that the device is active when the pin is high and above the threshold. If `active_state` is `False`, the device will be active when the pin is low and below the threshold.
- **threshold** (*float*) – The threshold that the device must be above or below to be considered active. The default is 0.5.
- **conversion** (*float*) –
  A function that takes a voltage and returns a temperature.
  e.g. The internal temperature sensor has a voltage range of 0.706V to 0.716V and would use the follow conversion function:

```
def temp_conversion(voltage):
    return 27 - (voltage - 0.706)/0.001721

temp_sensor = TemperatureSensor(pin, conversion=temp_conversion)
```

  If `None` (the default), the `temp` property will return `None`.

https://picozero.readthedocs.io/en/latest/api.html#temperaturesensor-tempsensor-thermistor

PicoZero Temp36

```python
from picozero import TemperatureSensor
from time import sleep

def TempCelsius(voltage):
    tempC = (100*voltage)-50
    tempC = round(tempC,1)
    return tempC

pin = 26
tmp36 = TemperatureSensor(pin, conversion=TempCelsius)

while True:
    tempC = tmp36.temp
    print(tempC, "°C")

    sleep(5)
```

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