



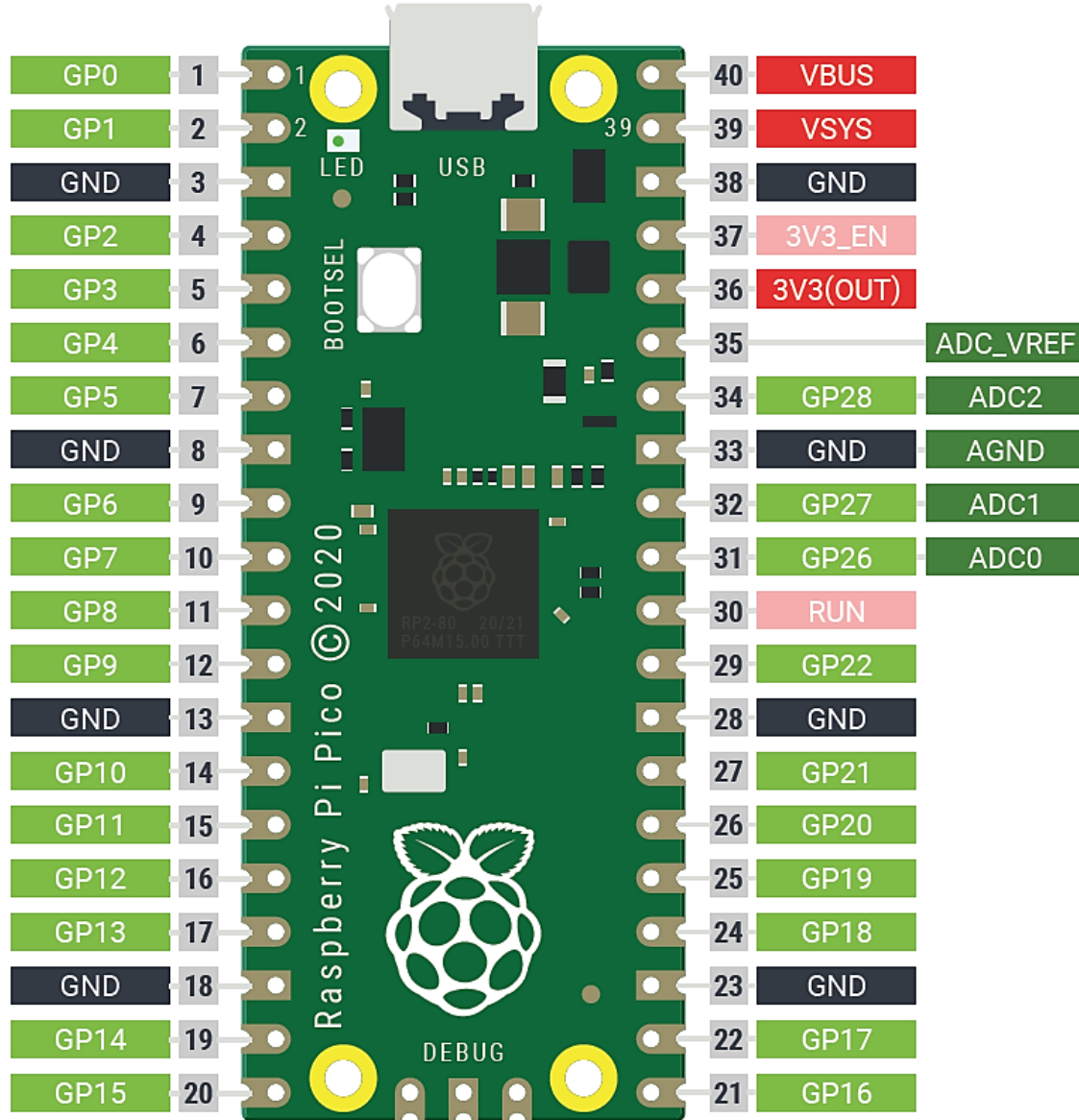
# Measuring temperatures using the Raspberry Pi Pico: inside the RP2040, not outside

Kruno Peter

Research and Teaching Department

Andrija Stampar Teaching Institute of Public Health

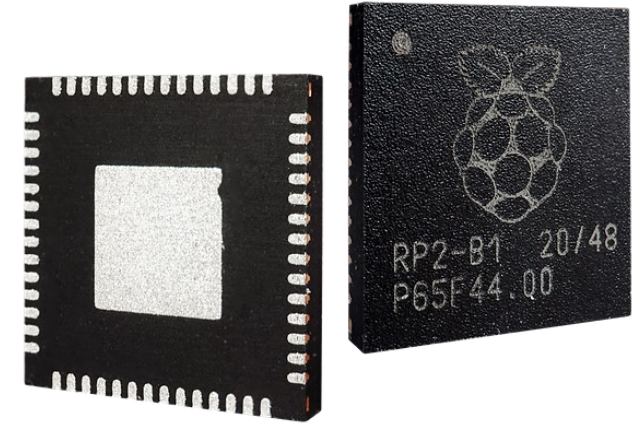
# the Raspberry Pi Pico microcontroller board\*



- the **RP2040** microcontroller:
  - dual-core Arm Cortex-M0+
  - 264 KB on-chip SRAM
  - a temperature sensor
- on-board
  - 2 MB flash memory
  - a micro-USB port
  - a green LED
- operating temperature: -20 °C to +85 °C
- programming:
  - C/C++
  - MicroPython
  - CircuitPython
  - Arduino IDE

\* Raspberry Pi Ltd 2022, 'Raspberry Pi Pico Datasheet – An RP2040-based microcontroller board'; Pico W and Pico H have different schemas.

# the internal temperature sensor<sup>#</sup>



- connected to the fifth channel of ADC<sup>#</sup>

```
sensor_temp = machine.ADC(4)
```

- Pico runs internally at 3.3 volts of power

- `read_u16()` reads an analogue input as a 16-bit integer<sup>#</sup>

```
conversion_factor = 3.3 / (65535)
```

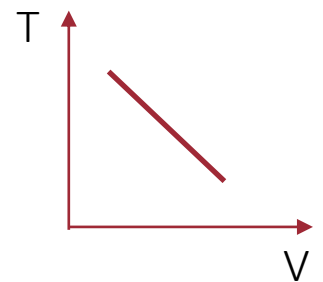
```
voltage = sensor_temp.read_u16() * conversion_factor
```

 3.3 V internally

- the sensor measures the  $V_{BE}$  voltage of a biased bipolar diode<sup>#</sup>

- typically,  $V_{BE} = 0.706$  V at 27 °C, with a slope of -1.721 mV (0.001721 V) per degree<sup>#</sup>

```
temperature = 27 - (voltage - 0.706) / 0.001721
```



## # added\*: visualizing temperatures and signaling a breach

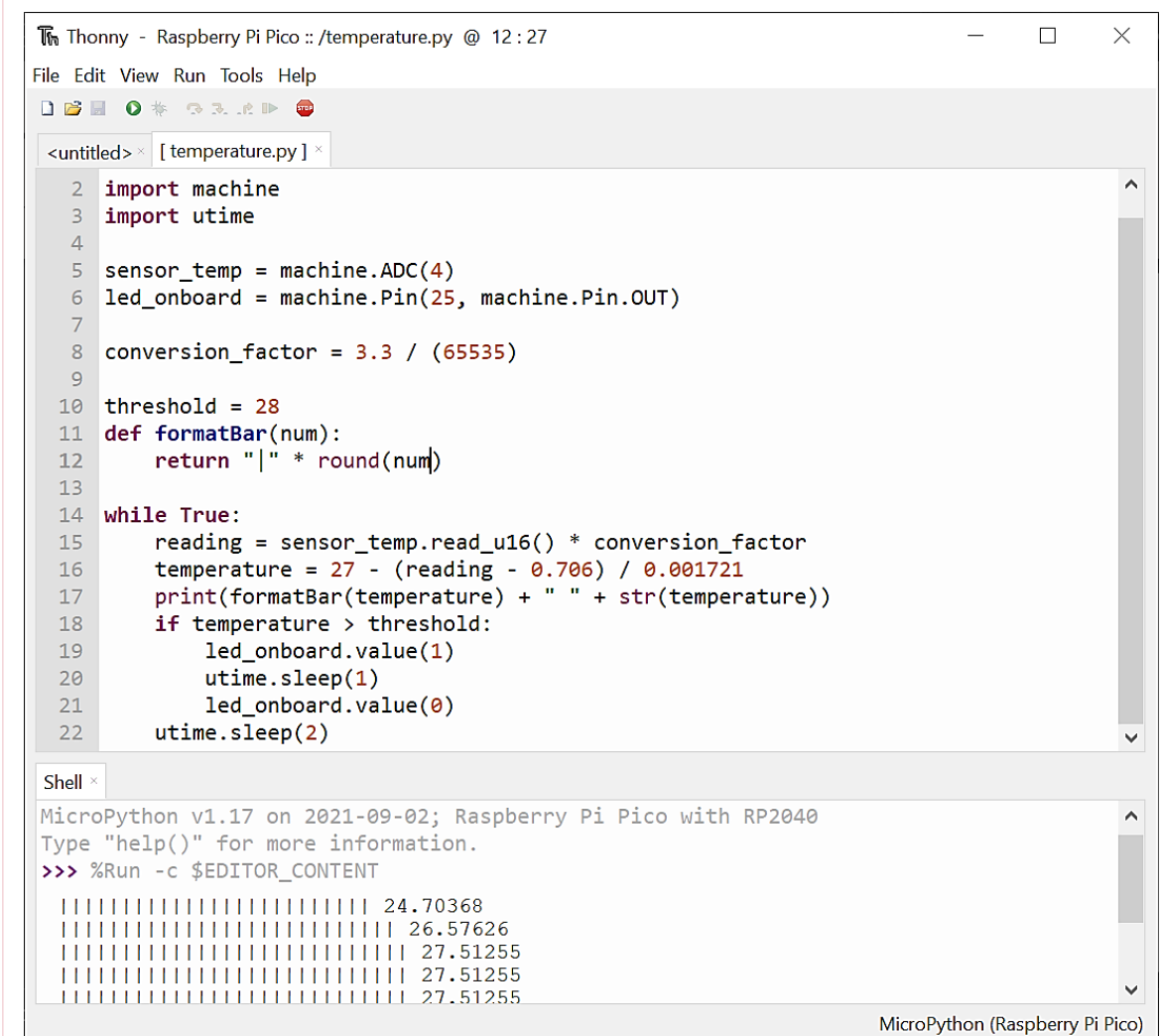
```
import machine
import utime
```

```
sensor_temp = machine.ADC(4)
led_onboard = machine.Pin(25, machine.Pin.OUT)
```

```
conversion_factor = 3.3 / (65535)
```

```
threshold = 28
def formatBar(num):
    return "|" * round(num)
```

```
while True:
    reading = sensor_temp.read_u16() * conversion_factor
    temperature = 27 - (reading - 0.706) / 0.001721
    print(formatBar(temperature) + " " + str(temperature))
    if temperature > threshold:
        led_onboard.value(1)
        utime.sleep(1)
        led_onboard.value(0)
        utime.sleep(2)
```



```
Thonny - Raspberry Pi Pico :: /temperature.py @ 12:27
File Edit View Run Tools Help

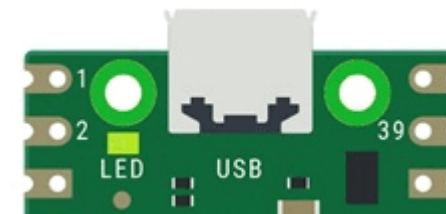
<untitled> x [ temperature.py ] x

2 import machine
3 import utime
4
5 sensor_temp = machine.ADC(4)
6 led_onboard = machine.Pin(25, machine.Pin.OUT)
7
8 conversion_factor = 3.3 / (65535)
9
10 threshold = 28
11 def formatBar(num):
12     return "|" * round(num)
13
14 while True:
15     reading = sensor_temp.read_u16() * conversion_factor
16     temperature = 27 - (reading - 0.706) / 0.001721
17     print(formatBar(temperature) + " " + str(temperature))
18     if temperature > threshold:
19         led_onboard.value(1)
20         utime.sleep(1)
21         led_onboard.value(0)
22         utime.sleep(2)

Shell x
MicroPython v1.17 on 2021-09-02; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

||||| 24.70368
||||| 26.57626
||||| 27.51255
||||| 27.51255
||||| 27.51255
||||| 27.51255

MicroPython (Raspberry Pi Pico)
```



## # signaling 2-digit temperatures by using the onboard LED\*

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

```
sensor_temp = machine.ADC(4)
led_onboard = Pin(25, Pin.OUT)
conversion_factor = 3.3 / (65535)
```

while True:

```
    voltage = sensor_temp.read_u16() * conversion_factor
    temperature = 27 - (voltage - 0.706) / 0.001721
    print("V = " + str(voltage) + "    T = " + str(temperature))
```

```
    digits = []
```

# extracting digits

```
    tens = abs(int(temperature / 10))
```

```
    digits.append(tens)
```

```
    ones = abs(round(temperature) - tens * 10)
```

```
    digits.append(ones)
```

```
    utime.sleep(6)
```

for x in digits:

# blinking every digit

```
    utime.sleep(3)
```

```
    for i in range(x):
```

```
        led_onboard.value(1)
```

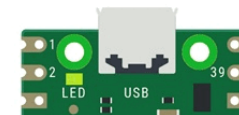

```
        utime.sleep(1)
```

```
        led_onboard.value(0)
```

```
        utime.sleep(1)
```

```
Thonny - C:\src\MicroPython\temperature_signalized.py @ 1:1
File Edit View Run Tools Help
temperature_signalized.py x
1 # signaling 2-digit temperatures by using the onboard LED
2 from machine import Pin, ADC
3 from utime import sleep
4
5 sensor_temp = ADC(4)
6 led_onboard = Pin(25, Pin.OUT)
7 conversion_factor = 3.3 / (65535)
8
9 while True:
10     voltage = sensor_temp.read_u16() * conversion_factor
11     temperature = 27 - (voltage - 0.706) / 0.001721
12     print("V = " + str(voltage) + "    T = " + str(temperature))
13     digits = [] # extracting digits
14     tens = abs(int(temperature / 10))
15     digits.append(tens)
16     ones = abs(round(temperature) - tens * 10)
17     digits.append(ones)
18     sleep(6)
19     for n in digits: # blinking every digit
20         sleep(3)
21         for i in range(n):
22             led_onboard.value(1)
23             sleep(1)
24             led_onboard.value(0)
25             sleep(1)
Shell x
>>> %Run -c $EDITOR_CONTENT
V = 0.7115633    T = 23.76739
V = 0.7115633    T = 23.76739
V = 0.7107576    T = 24.23554
MicroPython (Raspberry Pi Pico)
```

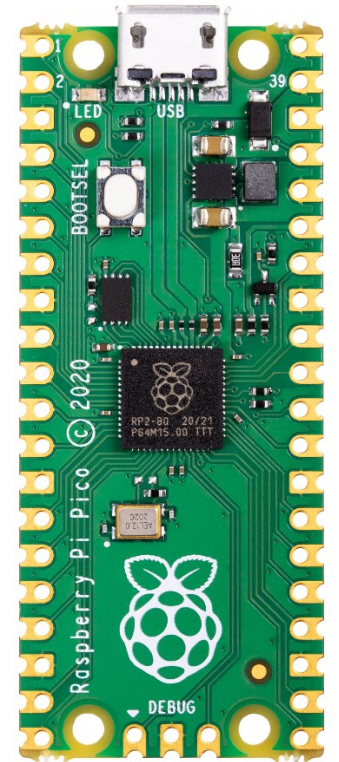
12 °C => 1 s



\* [https://github.com/kruno-peter/pico-temperature/blob/main/temperature\\_signalized.py](https://github.com/kruno-peter/pico-temperature/blob/main/temperature_signalized.py)

# the temperature inside the microcontroller and the ambient temperature

- is it possible to measure the environment temperature using the internal temperature sensor?
- Pico may report different temperatures than actual ambient temperatures
  - the microcontroller generates heat of its own\*
  - intensive data processing increases the internal temperature of the RP2040\*
- a solution: a temperature sensor, for example, the **DHT11** temperature & humidity sensor

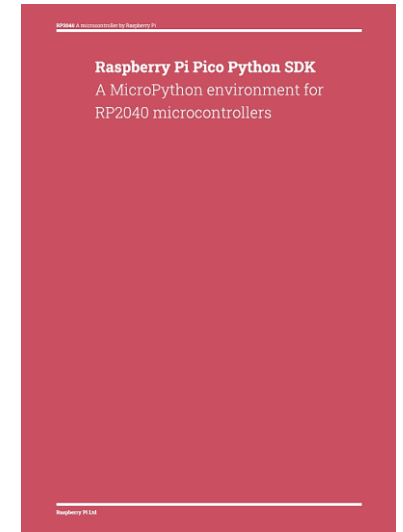
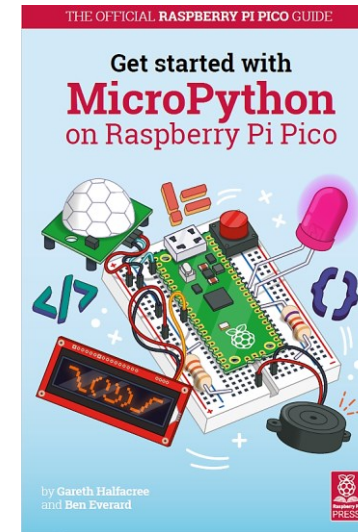
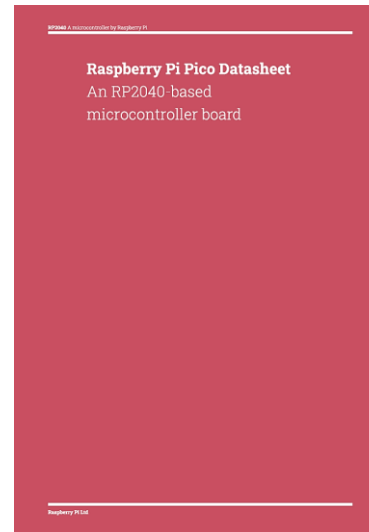
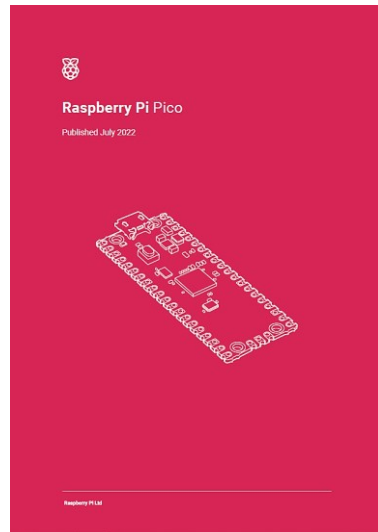




## conclusions

- the internal temperature sensor suitable for measuring temperatures inside RP2040 (not outside)
- documented and reusable MicroPython source code for that purpose

## literature



[raspberrypi.com](https://raspberrypi.com)



[micropython.org](https://micropython.org)

HS

[hackspace.raspberrypi.com](https://hackspace.raspberrypi.com)