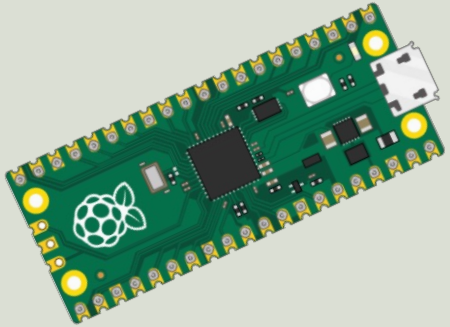
https://www.halvorsen.blog

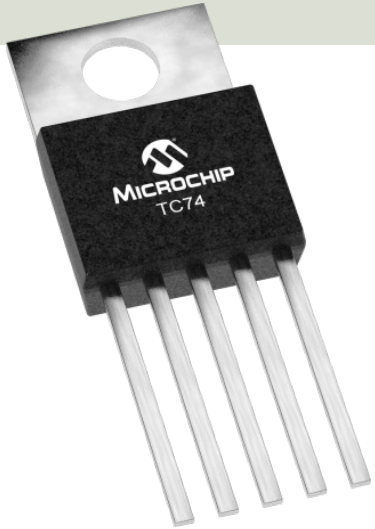


Raspberry Pi Pico

I2C Communication using TC74 Temperature Sensor

Hans-Petter Halvorsen

Contents



Introduction

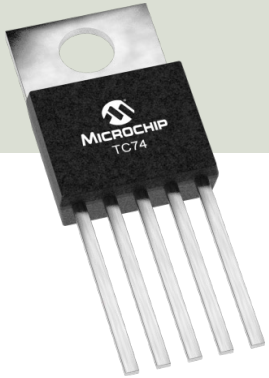
Hans-Petter Halvorsen

Table of Contents

Introduction

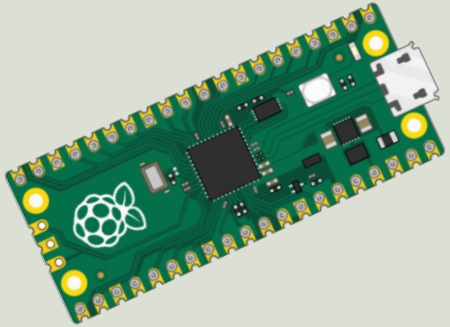
- In this Tutorial we will use a Raspberry Pi Pico and I2C Communication
- We will exemplify using a TC74 Temperature Sensor with I2C Interface
- We will use the Thonny Python Editor and MicroPython

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.
- **I2C Sensor**, we will use a **TC74** Temperature Sensor with I2C Interface in this Tutorial

https://www.halvorsen.blog



Raspberry Pi Pico

Hans-Petter Halvorsen

Table of Contents

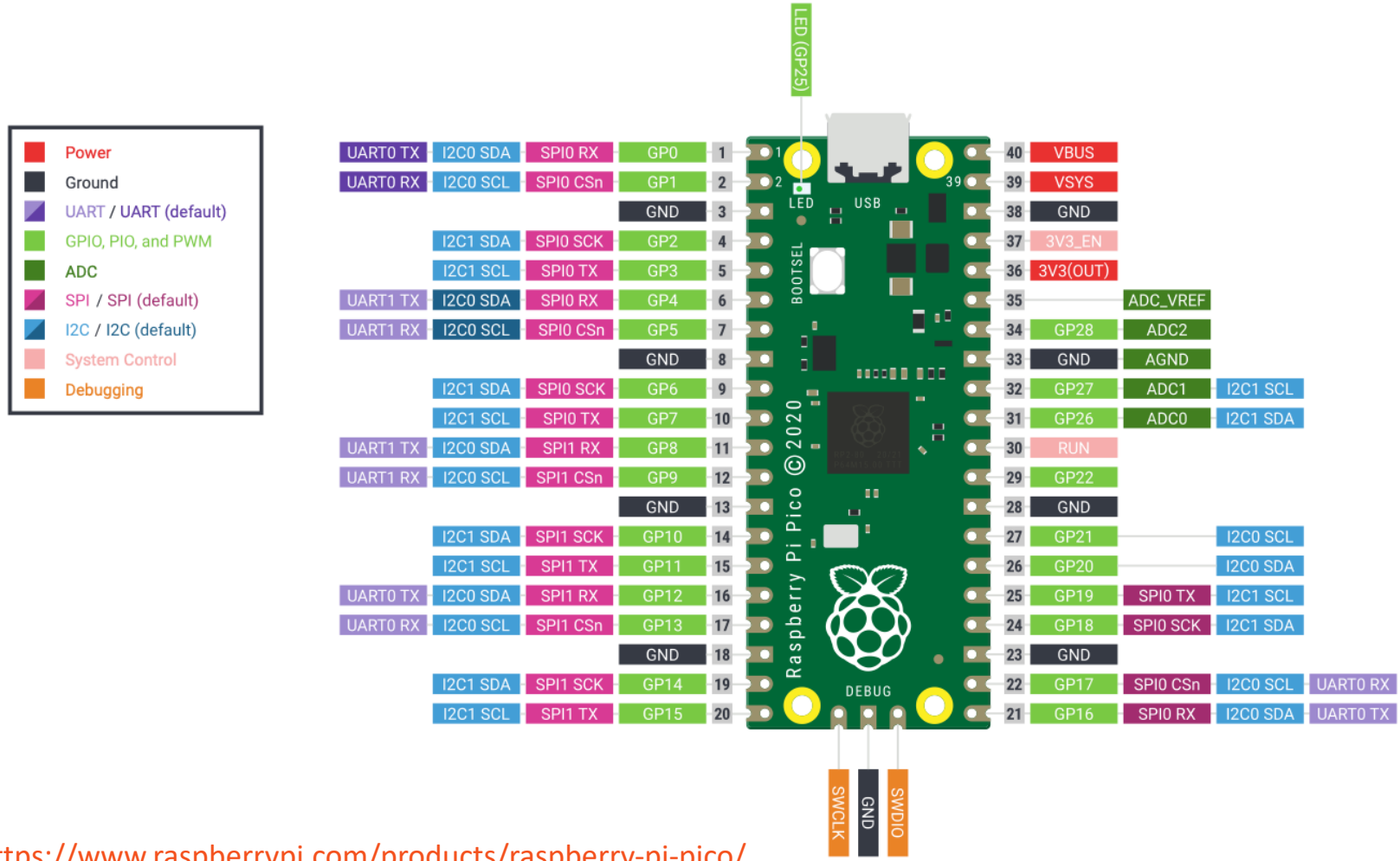
Raspberry Pi Pico

- 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/

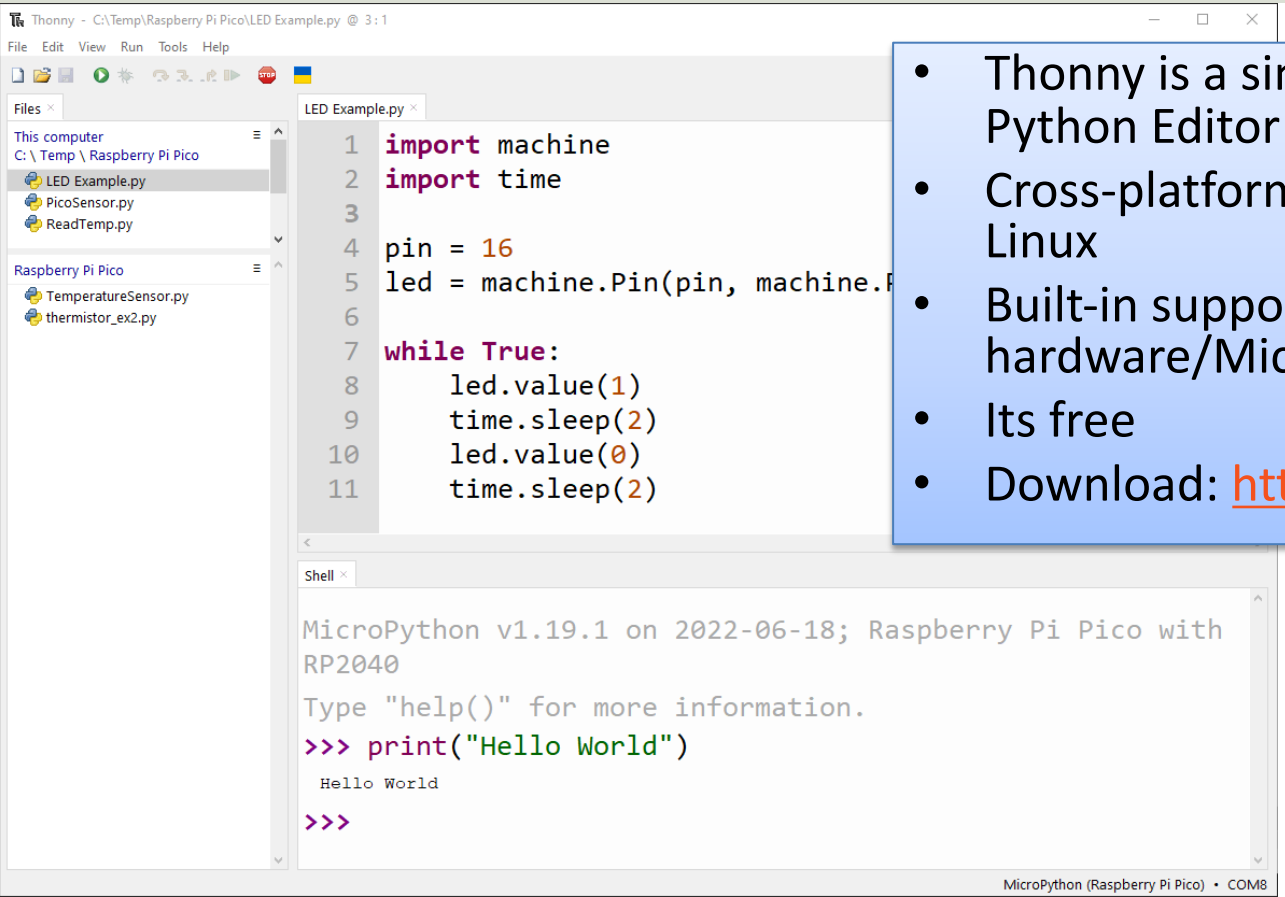
https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico

Pico Pinout



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

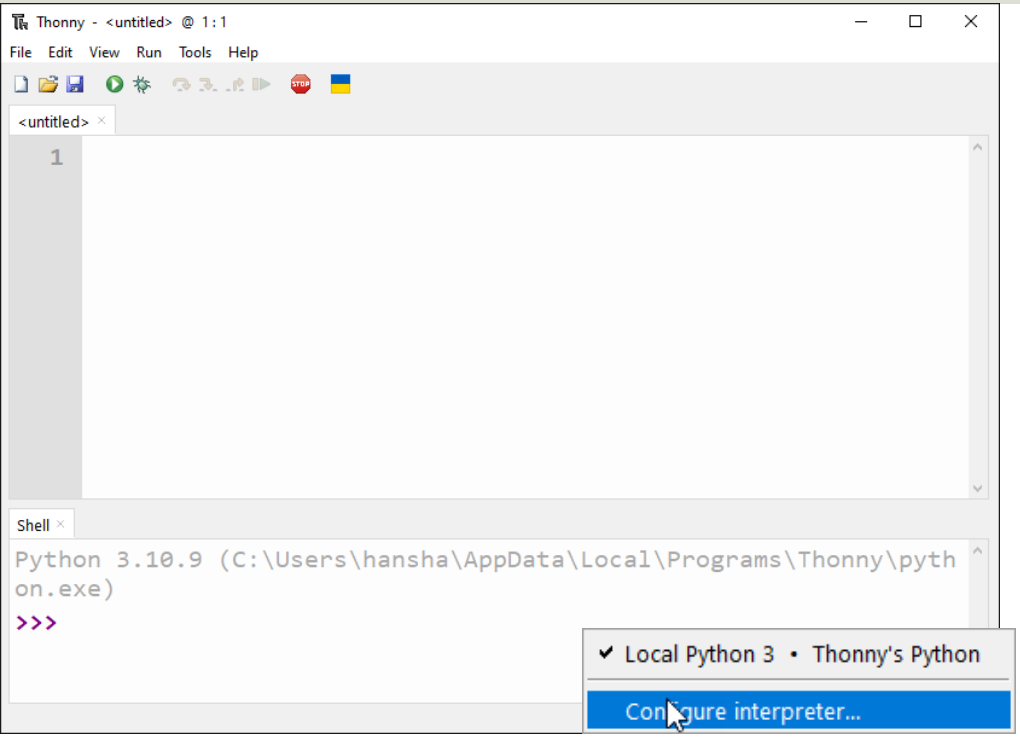
MicroPython

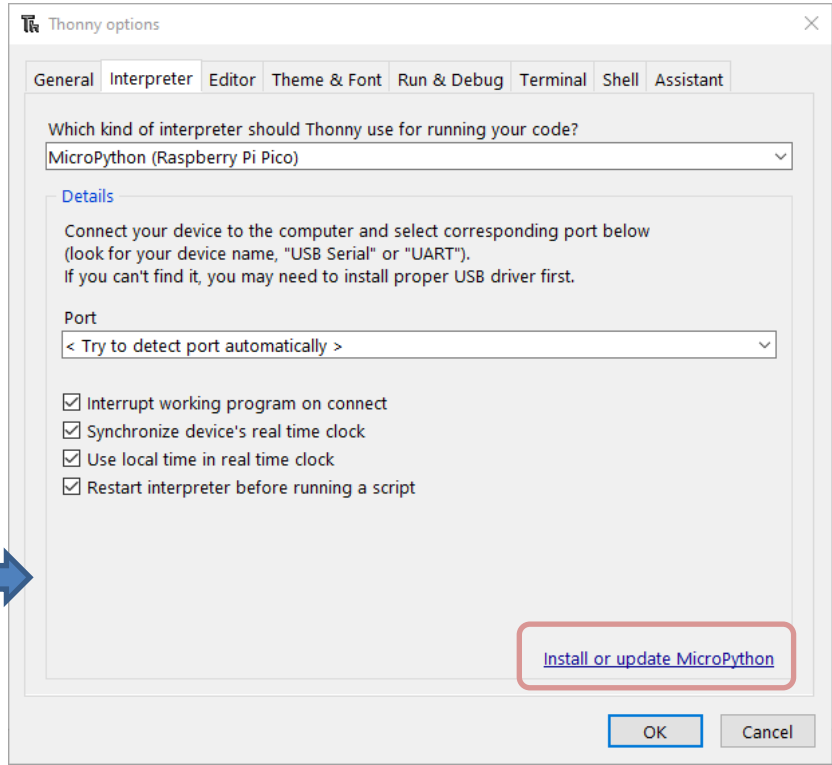
- **MicroPython is a downscaled version of Python**
- It is typically used for Microcontrollers and constrained systems (low memory, etc.)
- Examples of such Microcontrollers that have tailormade MicroPython firmware are Raspberry Pi Pico and Micro:bit
- https://micropython.org
- https://docs.micropython.org/en/latest/

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

I2C Communication

Inter-Integrated Circuit (I^2C)

Hans-Petter Halvorsen

Table of Contents

I2C

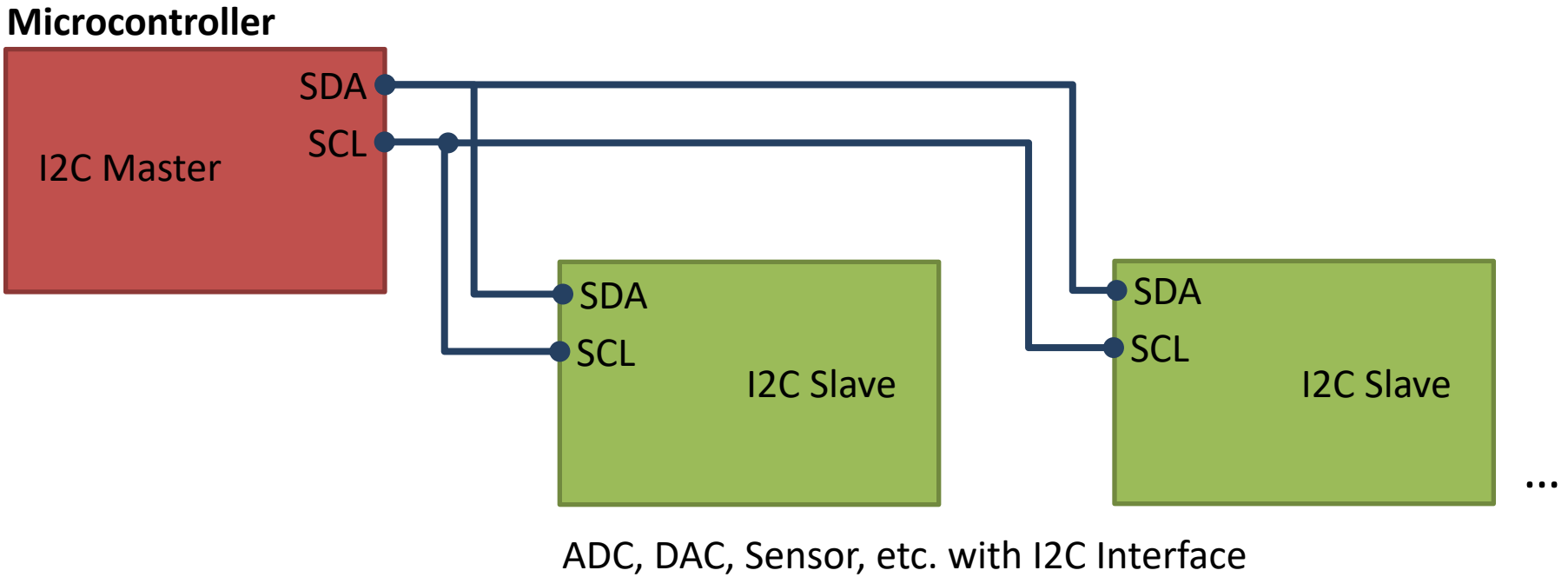
- With the I2C protocol you can communicate using just two wires, a clock and data line (+ Power and GND)
- Typically you use I2C to talk to devices like sensors, small displays, PWM or motor drivers, and other devices.
- The Sensor you want to communicate with needs to support the I2C protocol
- There exist thousands of different Sensors, etc. that support the I2C Protocol
- Most Microcontrollers today supports I2C Communication

I2C

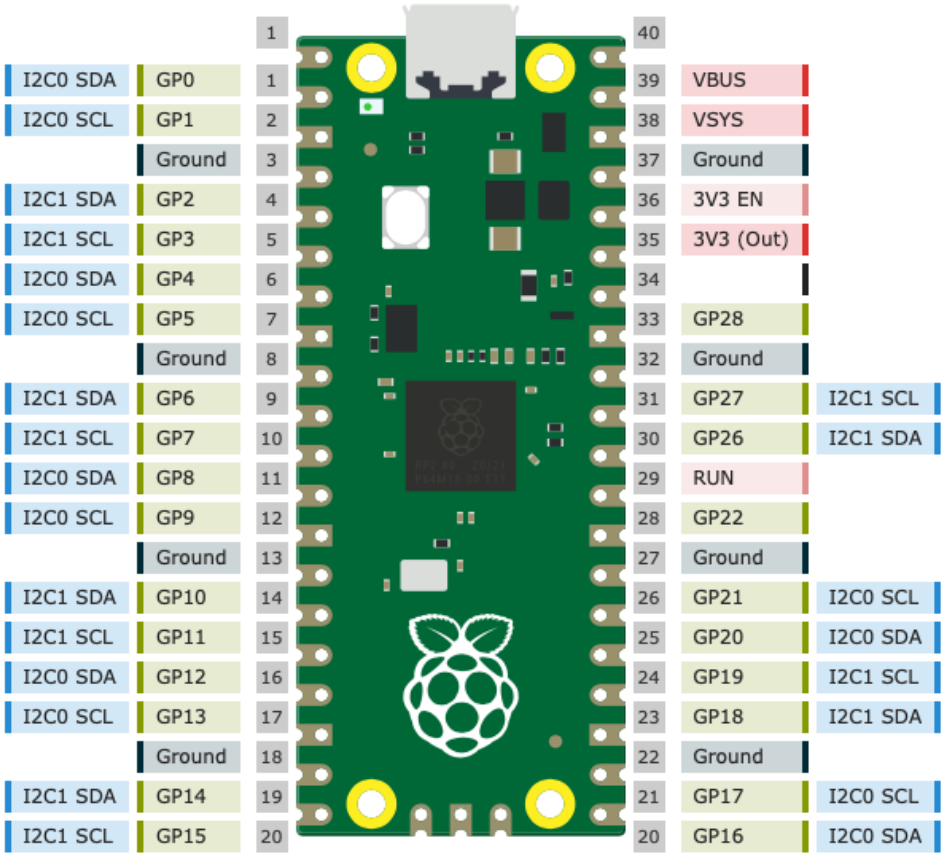
- I2C is a multi-drop bus
- 2-Wire Protocol: SCL (Clock) + SDA (Data)
- Multiple devices can be connected to the I2C pins on the Raspberry Pi Pico
- Each device has its own unique I2C address

I2C

Multiple devices can be connected to the I2C pins on the Arduino
Master – Device that generates the clock and initiates communication with slaves
Slave – Device that receives the clock and responds when addressed by the master.



I2C with Pico



- **Raspberry Pi Pico has 2 I2C Controllers (0 and 1)**.
- You can access these I2C controllers through most of the GPIO pins of Raspberry Pi Pico.
- So, you should configure in **software** (your MicroPython program) which GPIO pins you want to use with a specific I2C controller.

I2C with Pico

```
from machine import I2C
```

Initialize I2C Communication: Raspberry Pi Pico has 2 I2C Controllers/Interfaces (0 and 1)

```
i2c = I2C(i2c_interface, scl=sclpin, sda=sdapin, freq=100000)
```

"freq" should be an integer which sets the maximum frequency for SCL

Read Data from the connected I2C device:

```
data = i2c.readfrom(address, n, True)
```

Read n bytes from the peripheral specified by address. If True is set, then a STOP condition is generated at the end of the transfer. The function returns a bytes object with the data.

Many other I2C functions do exist, see documentation :

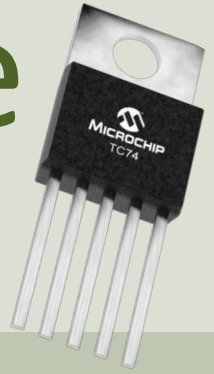
https://docs.micropython.org/en/latest/library/machine.I2C.html

Python – Scan for I2C Devices

```
from machine import Pin, I2C

i2c_interface = 0

sdapin = Pin(16)
sclpin = Pin(17)

i2c = I2C(i2c_interface, scl=sclpin, sda=sdapin, freq=100000)

i2cdevices = i2c.scan()

print(i2cdevices)
```

https://www.halvorsen.blog

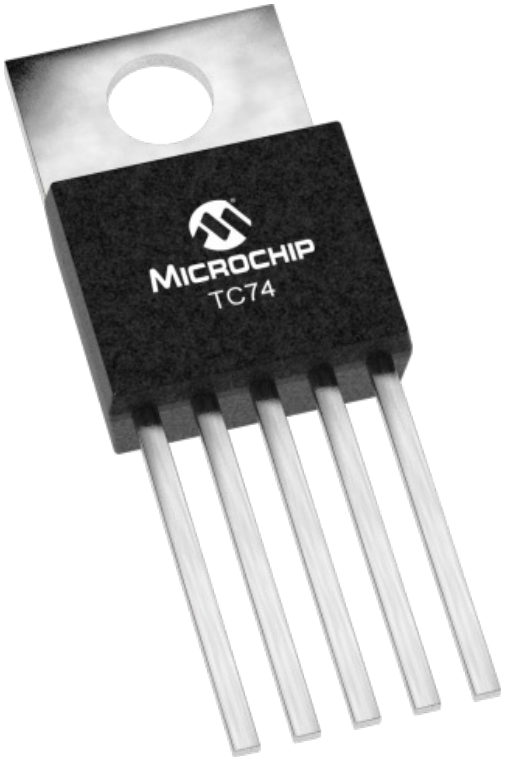
TC74 Temperature Sensor with I2C Interface

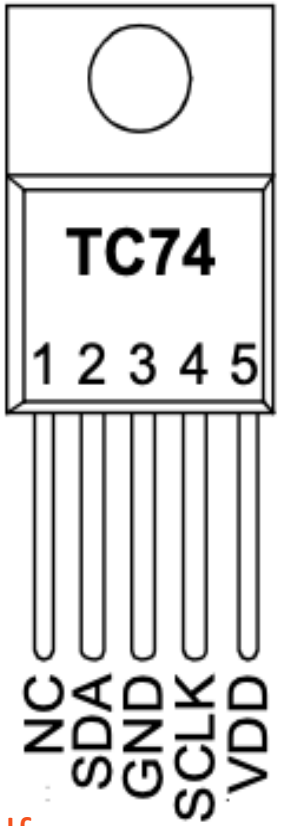


Hans-Petter Halvorsen

Table of Contents

TC74 Temperature Sensor





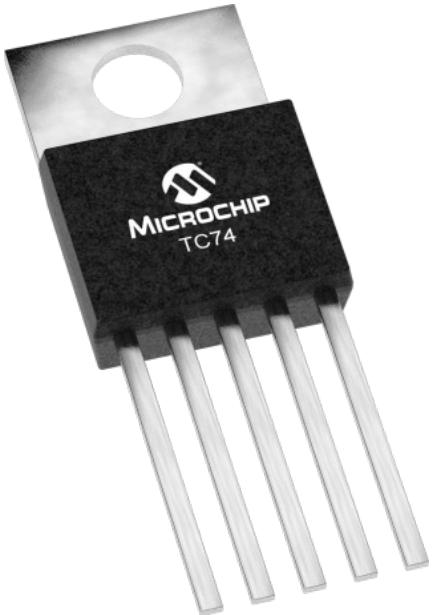
Make sure to buy the breadboard friendly **TO-220** package version of the sensor

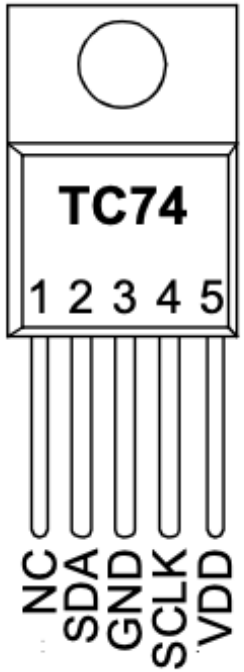
Datasheet: https://ww1.microchip.com/downloads/en/DeviceDoc/21462D.pdf

TC74 Temperature Sensor

SMBus/I2C Interface

TC74A0-5.0VAT

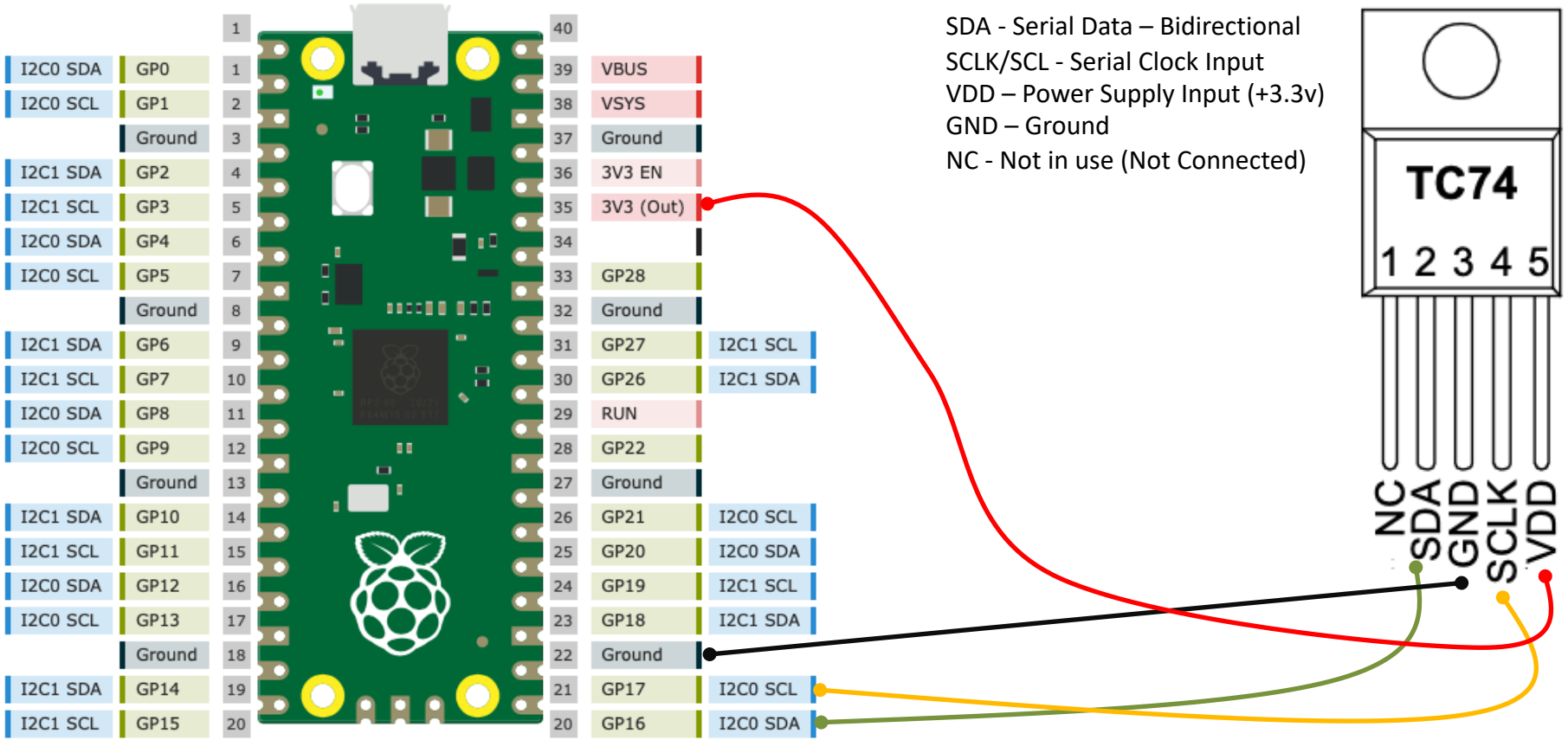




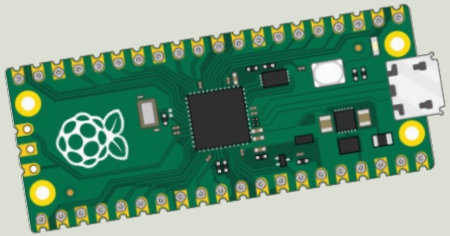
- The TC74 acquires and converts temperature information from its onboard solid-state sensor with a **Resolution of 1°C** (no decimal values, only 24°C, 25 °C, 26 °C, etc.).
- **Accuracy** is about ±2°C
- It stores the data in an internal register which is then read through the serial port.
- The system interface is a slave SMBus/I2C port, through which temperature data can be read at any time.
- Device Address: **0x48**

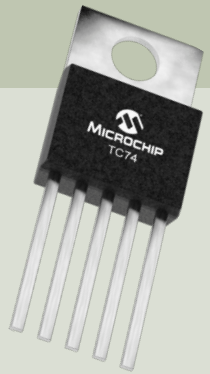
Datasheet: https://ww1.microchip.com/downloads/en/DeviceDoc/21462D.pdf

TC74 Wiring Example



https://www.halvorsen.blog





Python Examples using MicroPython and TC74

Hans-Petter Halvorsen

Table of Contents

Python

Basic Example reading a Temperature Value from the TC74 Temperature Sensor

```
from machine import Pin, I2C

i2c_interface = 0
sdapin = Pin(16)
sclpin = Pin(17)

i2c =I2C(i2c_interface, scl=sclpin, sda=sdapin,
freq=100000)

tc74address = 0x48
data = i2c.readfrom(tc74address, 1, True)
print(data)

temp = int.from_bytes(data, "big")
print(temp)
```

Python

Basic Example reading a Temperature Value from the TC74 Temperature Sensor

```
from machine import Pin, I2C

i2c_interface = 0
sdapin = Pin(16)
sclpin = Pin(17)

i2c =I2C(i2c_interface, scl=sclpin, sda=sdapin, freq=100000)

tc74address = 0x48
data = i2c.readfrom(tc74address, 1, True)
print(data)  # Data received is a byte object

# Converting to int. Resolution for TC74 Sensor is 1°C
# byteorder is big where MSB is at start
temp = int.from_bytes(data, "big")
print(temp)
```

Code Explanations

```
..
data = i2c.readfrom(tc74address, 1, True)
```

Temperature data is available as an 8-bit digital word.

We need to convert the data from a byte array to an Integer value. Resolution for TC74 Sensor is 1°C, meaning there is no decimal values only 24°C, 25 °C, 26 °C, etc.

```
temp = int.from_bytes(data, "big")
```

The byteorder argument determines the byte order used to represent the integer. **If byteorder is "big", the most significant byte is at the beginning of the byte array.** If byteorder is "little", the most significant byte is at the end of the byte array

Thonny - C:\Users\hansha\OneDrive\Documents\Industrial IT and Automation\IoT\Raspberry Pi Pico\Code Examples\i2c_tc74.py @ 1 : 29

File Edit View Run Tools Help

i2c_scan.py i2c_tc74.py

```
1  from machine import Pin, I2C
2
3  i2c_interface = 0
4
5  sdapin = Pin(16)
6  sclpin = Pin(17)
7
8  i2c =I2C(i2c_interface, scl=sclpin, sda=sdapin, freq=100000)
9
10 tc74address = 0x48
11
12 data = i2c.readfrom(tc74address, 1, True)
13 print(data)  # Data received is a byte object
14
15 # Converting to int. Resolution for TC74 Sensor is +/-1°C
16 # byteorder is big where MSB is at start
17 temp = int.from_bytes(data, "big")
18 print(temp)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
  b'\x1a'
  26
>>>
```

The Temperature in this case is 26°C

MicroPython (Raspberry Pi Pico) • COM6

Continues Reading Example

```
from machine import Pin, I2C
from time import sleep

i2c_interface = 0

sdapin = Pin(16)
sclpin = Pin(17)

i2c = I2C(i2c_interface, scl=sclpin, sda=sdapin,
freq=100000)

tc74address = 0x48

while True:
    data = i2c.readfrom(tc74address, 1, True)
    temp = int.from_bytes(data, "big")
    print(temp)
    sleep(5)
```

Thonny - C:\Users\hansha\OneDrive\Documents\Industrial IT and Automation\IoT\Raspberry Pi Pico\Code Examples\i2c_tc74v2.py @ 10 : 1

File Edit View Run Tools Help

i2c_scan.py | i2c_tc74.py | i2c_tc74v2.py

```
1  from machine import Pin, I2C
2  from time import sleep
3
4  i2c_interface = 0
5
6  sdapin = Pin(16)
7  sclpin = Pin(17)
8
9  i2c =I2C(i2c_interface, scl=sclpin, sda=sdapin, freq=100000)
10
11 tc74address = 0x48
12
13 while True:
14     data = i2c.readfrom(tc74address, 1, True)
15     temp = int.from_bytes(data, "big")
16     print(temp)
17     sleep(5)
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
 27
 27
 27
 28
 29
 27
 27
 28
 27
 28
 27
 28
 28
```

MicroPython (Raspberry Pi Pico) • COM6

Improved Example

Let's make a separate Python Module with a Class and a Function that handles all the logic regarding reading Temperature Data from the TC74 Temperature Sensor

Sensor.py

```
from machine import Pin, I2C

class TC74:
    def __init__(self, interface, sda, scl):
        sdapin = Pin(sda)
        sclpin = Pin(scl)
        self.i2c = I2C(interface, scl=sclpin, sda=sdapin, freq=100000)

    def ReadTemperature(self):
        tc74address = 0x48
        data = self.i2c.readfrom(tc74address, 1, True)
        temp = int.from_bytes(data, "big")
        return temp
```

Main Program

```
from Sensor import TC74
from time import sleep

# Initialization
i2c_interface = 0
sdapin = 16; sclpin = 17
sensor = TC74(i2c_interface, sdapin, sclpin)

while True:
    temp = sensor.ReadTemperature()
    print(temp, "°C")
    sleep(5)
```

Datalogging and Data Analysis Examples

Hans-Petter Halvorsen

Table of Contents

Datalogging

- We will read data from a Temperature Sensor using
- We will then Log Temperature Data on a File on the Pico Device
- Then we will copy the File to our PC and are then ready to do some Data Analysis
- Finally, we will create a simple Python Script that opens the File and Plot the Data. Here we will use ordinary Python and the matplotlib

```
from machine import Pin, I2C
from time import sleep

#I2C Initialization
tc74address = 0x48
i2c_interface = 0
sdapin = Pin(16); sclpin = Pin(17)
i2c =I2C(i2c_interface, scl=sclpin, sda=sdapin, freq=100000)

# Open File
file = open("tempdata.txt", "w")

# Write Data to File Function
def writefiledata(t, x):
    time = str(t)
    value = str(round(x, 2))
    file.write(time + "\t" + value)
    file.write("\n")

k = 0
Ts = 5
while True:
    data = i2c.readfrom(tc74address, 1, True)
    temp = int.from_bytes(data, "big")
    print(temp)
    writefiledata(k*Ts, temp)
    k = k + 1
    sleep(Ts)
```

Thonny - Raspberry Pi Pico :: /tempdata.txt @ 6 : 6

File Edit View Run Tools Help

Files

This computer
C: \ Users \ hansha

3D Objects

Raspberry Pi Pico

lib

tempdata.txt

i2c_tc74_datalogging.py [tempdata.txt] *

```
1  0   27
2  5   28
3  10  26
4  15  26
5  20  27
6  25  28
```

Shell

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

  27
  28
  26
  26
  27
  28


  Traceback (most recent call last):
    File "<stdin>", line 29, in <module>
  KeyboardInterrupt:
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

MicroPython (Raspberry Pi Pico) • COM6

Data Analysis

- The File is now copied to our PC and we are then ready to do some Data Analysis
- We will create a simple Python Script that opens the File and Plot the Data. Here we will use ordinary Python and the matplotlib

```
import matplotlib.pyplot as plt

# Open File
f = open("tempdata.txt", "r")

# Transform File Data into x Array and y Array that can be used for plotting
x = []
y = []
k = 0
for record in f:
    record = record.replace("\n", "")
    record = record.split("\t")
    x.append(int(record[0]))
    y.append(int(record[1]))
    k=k+1

f.close()

plt.plot(x,y, '-o')
plt.title('Temperature Data from TC74 Sensor')
plt.xlabel('Time[s]')
plt.ylabel('Temperature[°C]')
plt.grid()
plt.show()
```

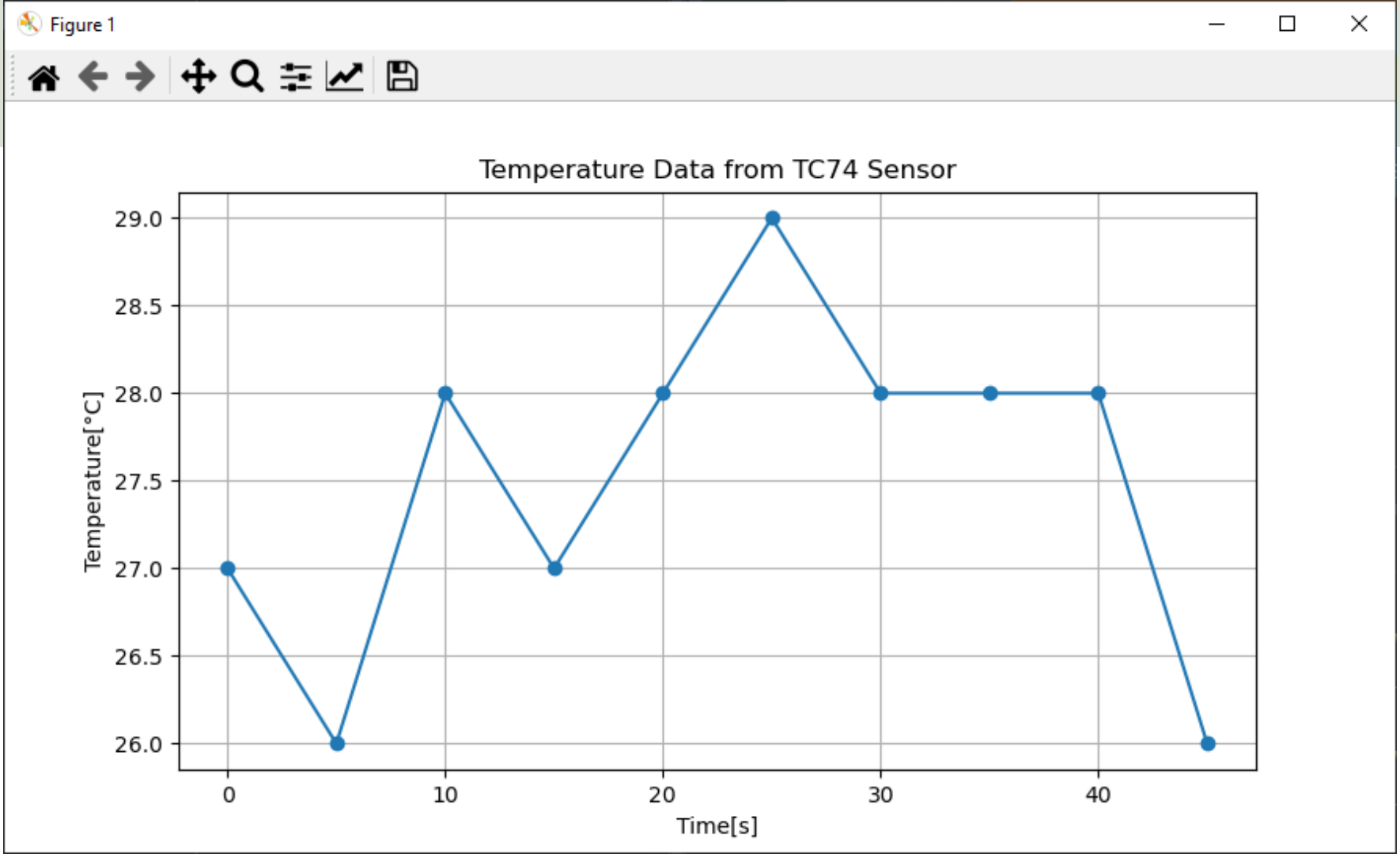


Figure 1
Temperature Data from TC74 Sensor
29.0
28.5
28.0
27.5
27.0
26.5
26.0
Temperature[°C]
0
10
20
30
40
Time[s]

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