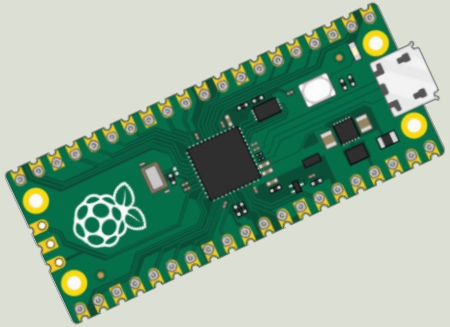
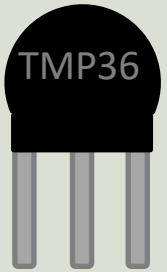
https://www.halvorsen.blog



Raspberry Pi Pico

Logging Sensor Data



Hans-Petter Halvorsen

Contents

Introduction

Hans-Petter Halvorsen

Table of Contents

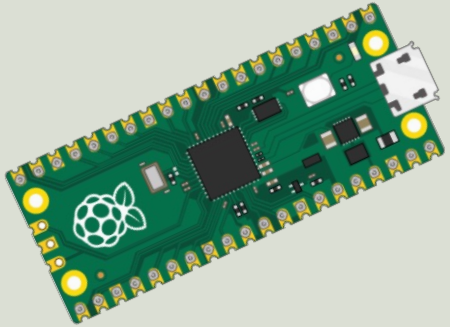
Introduction

- In this Tutorial we will log data from a Temperature Sensor using MicroPython.
  - We will use a basic TMP36 Temperature Sensor
- We will Log Temperature Data on a File on the Raspberry Pi Pico Device.
- Then we will copy the File to our PC and 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.

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.
- **Sensor**, we will use a **TMP36** Temperature Sensor in this Tutorial

https://www.halvorsen.blog



Raspberry Pi Pico

Hans-Petter Halvorsen

Table of Contents

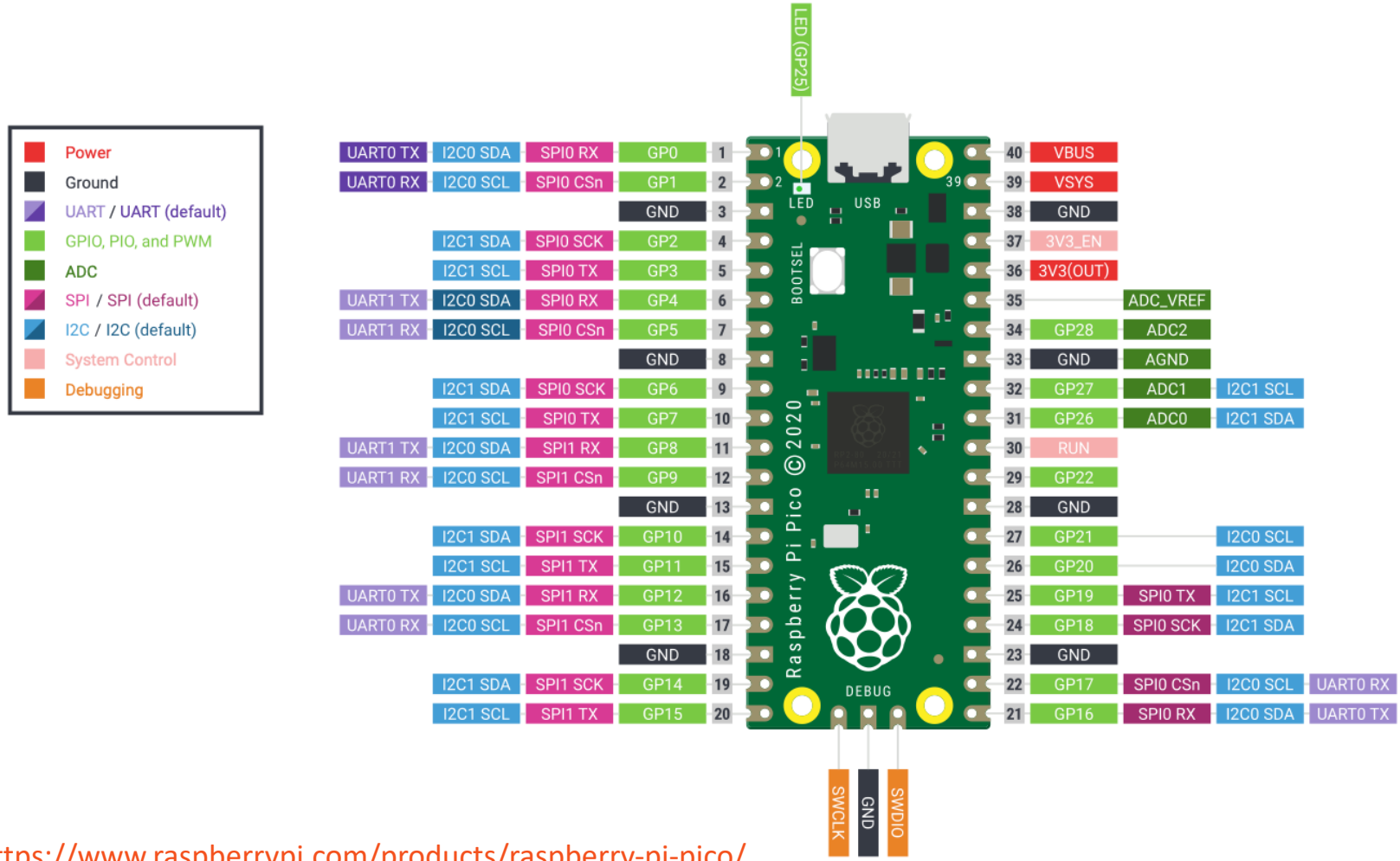
Raspberry Pi Pico

- 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/

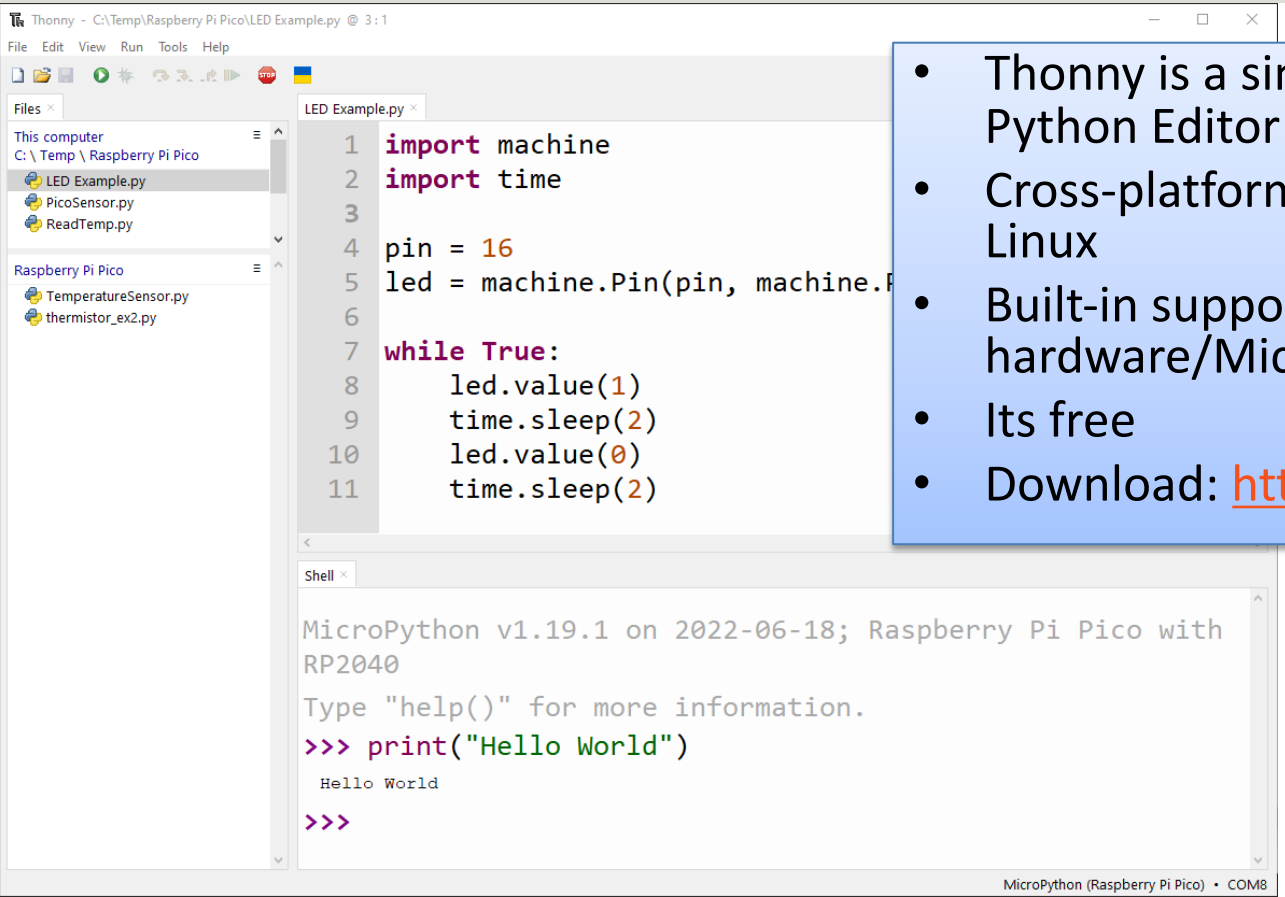
https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico

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

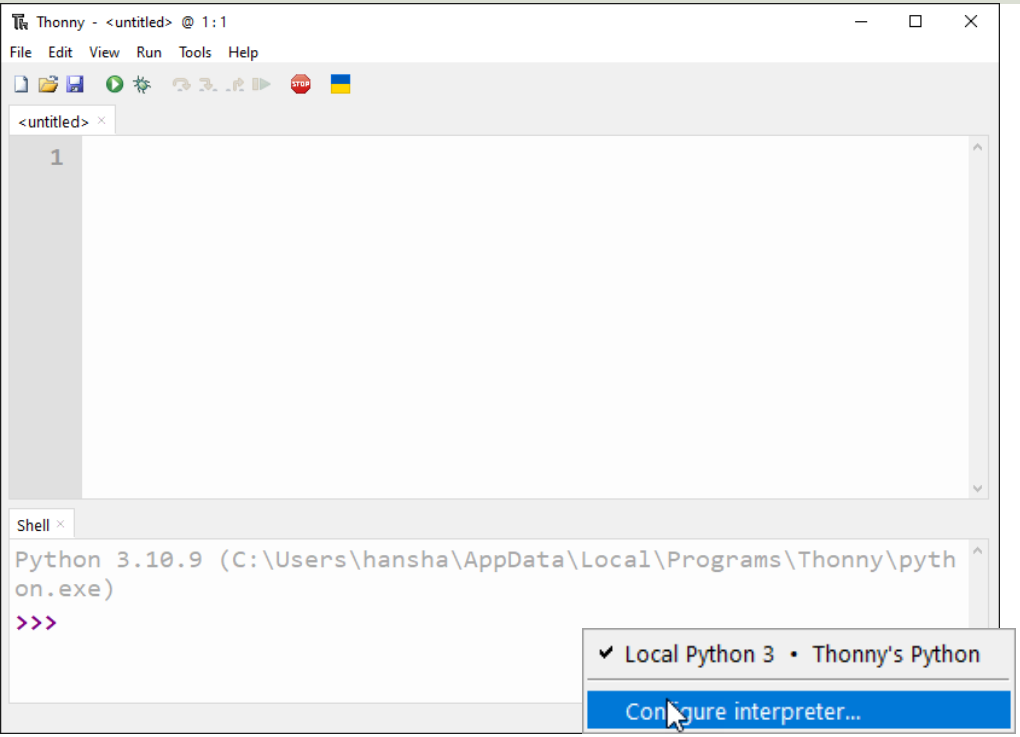
MicroPython

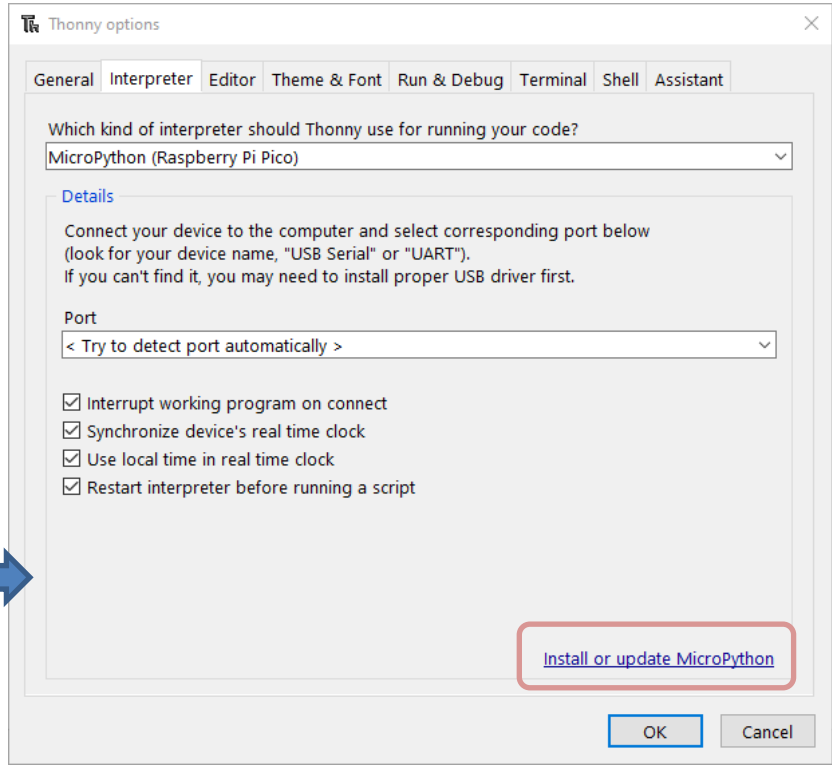
- MicroPython is a downscaled version of Python
- It is typically used for Microcontrollers and constrained systems (low memory, etc.)
- Examples of such Microcontrollers that have tailormade MicroPython firmware are Raspberry Pi Pico and Micro:bit
- https://micropython.org
- https://docs.micropython.org/en/latest/

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

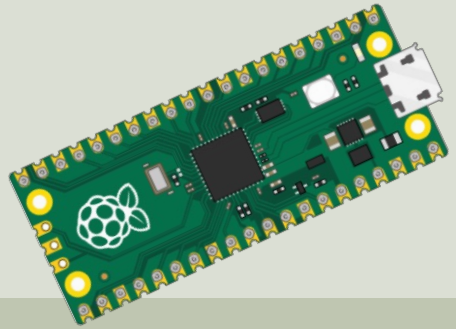


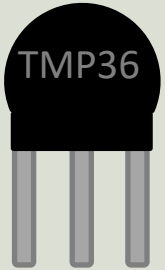


https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/3

https://www.halvorsen.blog

TMP36 Temperature Sensor

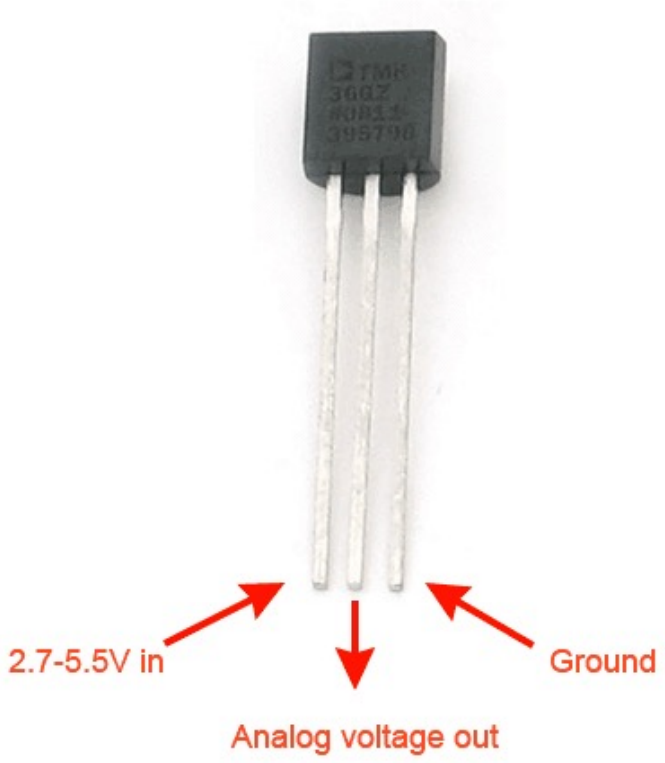




Hans-Petter Halvorsen

Table of Contents

TMP36 Temperature Sensor

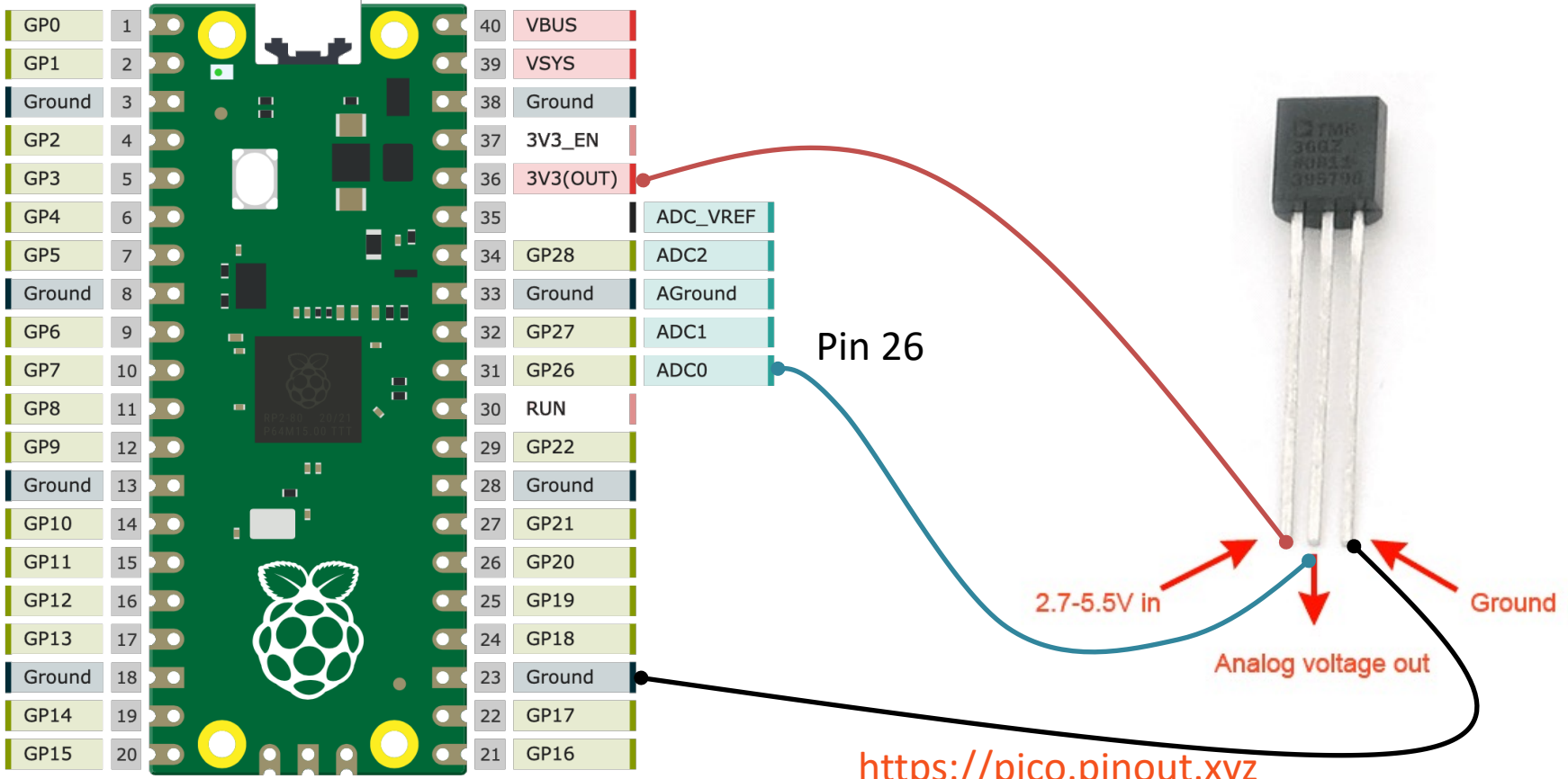


A Temperature sensor like TM36 use a solid-state technique to determine the temperature.

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

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

TMP36 Wiring

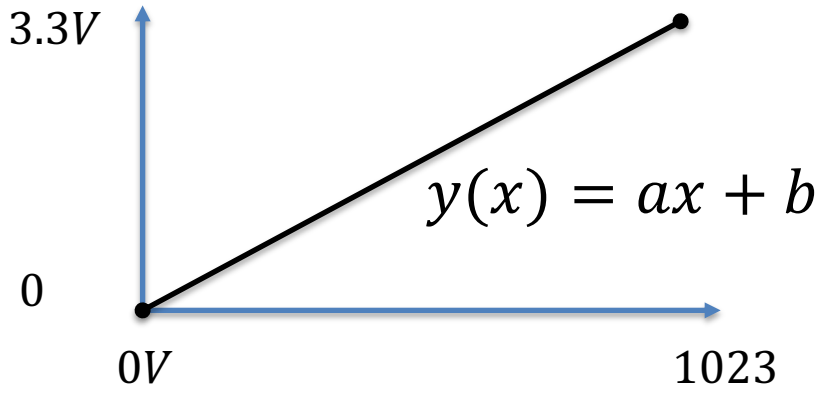


ADC Value to Voltage Value

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

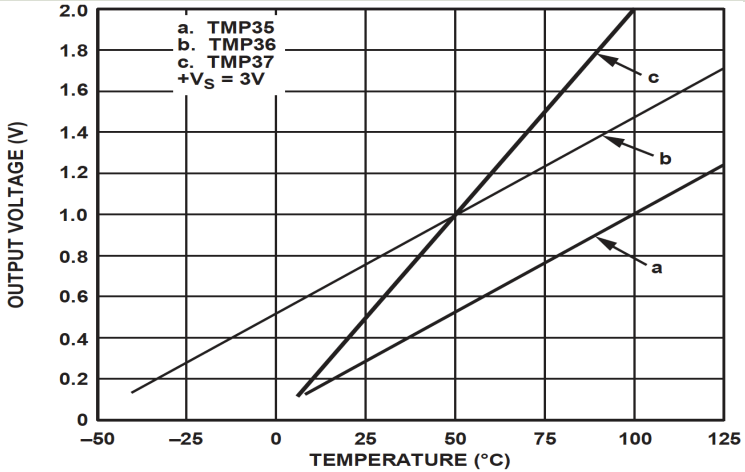
ADC = 0 -> 0v
ADC = 65535 -> 3.3v



This gives the following conversion formula:

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

Voltage to degrees Celsius



This gives:

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

Then we get the following formula:

$$y = 100x - 50$$

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:

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

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

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

```
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)
```

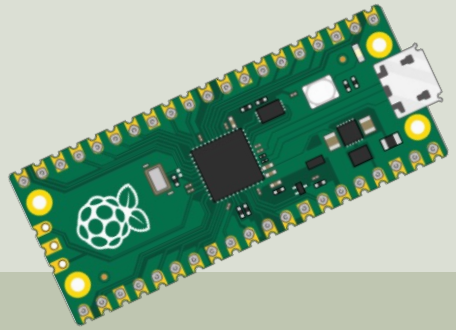
Shell

```
>>> %Run -c $EDITOR_CONTENT
  25.7
  25.6
  27.5
  30.3
  28.8
  27.2
  26.8
  26.7
```

MicroPython (Raspberry Pi Pico) • COM6

https://www.halvorsen.blog

Datalogging and Analysis



Hans-Petter Halvorsen

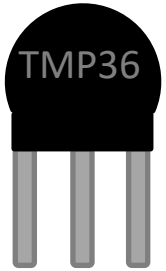
Table of Contents

Datalogging and Analysis

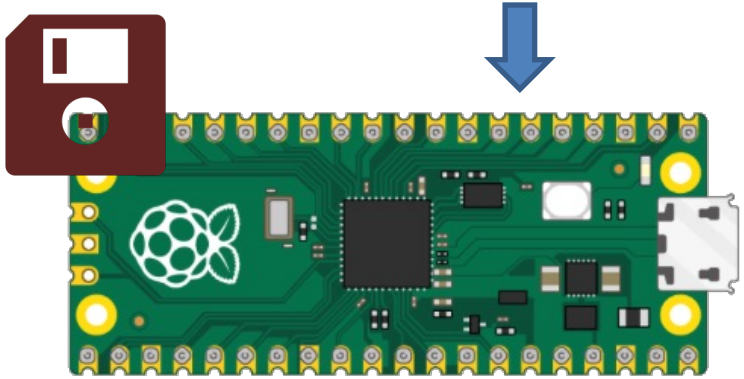
- We will read data from a Temperature Sensor using Raspberry Pi Pico and MicroPython
- We will then Log Temperature Data on a File on the Pico Device
- Then we will copy the File to our PC and are then ready to do some Data Analysis
- Finally, we will create a simple Python Script that opens the File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

Datalogging and Analysis

1. Read Sensor Data



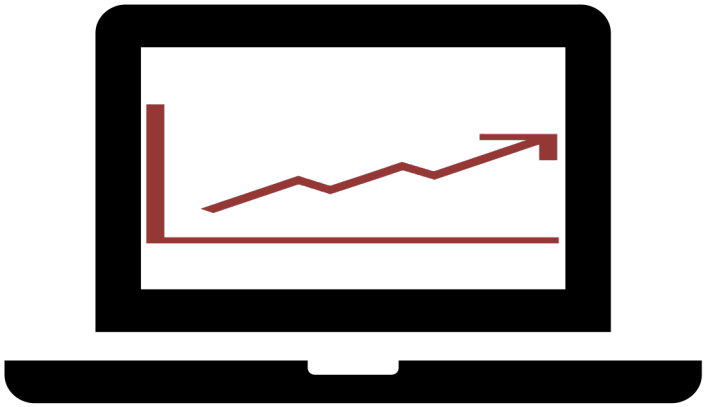
2. Save Data to local File



Read Data from Sensor and store on local File on Raspberry Pi Pico

Copy Data File to PC for Data Analysis

3. Data Analysis



Plotting Data, Calculate Statistics, create Data Models, etc.

File Handling in Python

Hans-Petter Halvorsen

Table of Contents

File Handling in Python

Write Data to a File:

```
file = open("data.txt", "w")
```

```
file.write("Hello World")
```

```
file.close()
```

"w" - write
Use "a" (append) if you don't want to delete existing Data

Read Data from a File:

```
file = open("data.txt", "r")
```

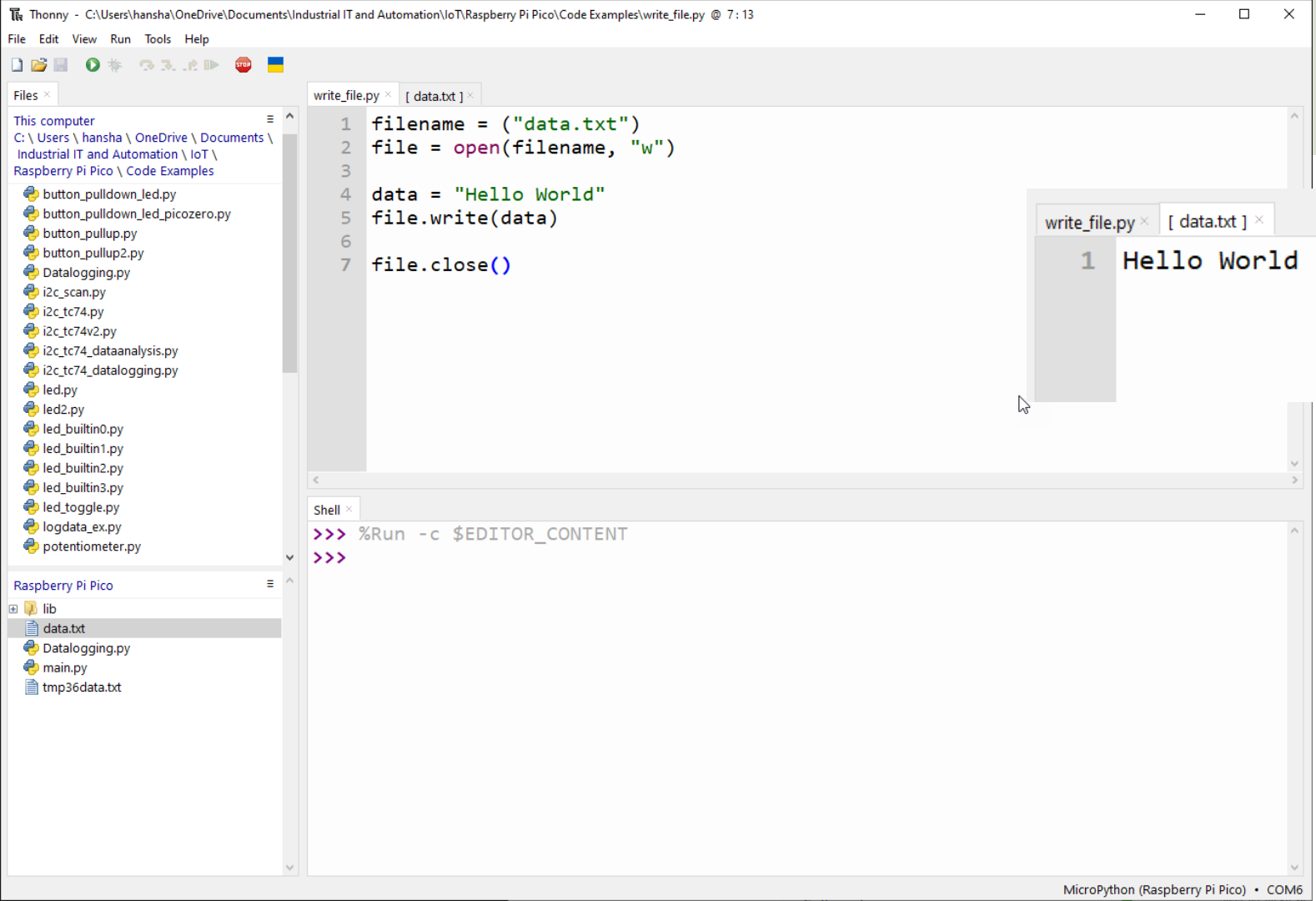
```
data = file.read()
```

```
file.close()
```

"r" - read

Open and Write Data to File in Python

```
filename = "data.txt"
file = open(filename, "w")

data = "Hello World"
file.write(data)

file.close()
```


Thonny - C:\Users\hansha\OneDrive\Documents\Industrial IT and Automation\IoT\Raspberry Pi Pico\Code Examples\write_file.py @ 7 : 13
File Edit View Run Tools Help
Files
This computer
C: \ Users \ hansha \ OneDrive \ Documents \ Industrial IT and Automation \ IoT \ Raspberry Pi Pico \ Code Examples
button_pulldown_led.py
button_pulldown_led_picozero.py
button_pullup.py
button_pullup2.py
Datalogging.py
i2c_scan.py
i2c_tc74.py
i2c_tc74v2.py
i2c_tc74_dataanalysis.py
i2c_tc74_datalogging.py
led.py
led2.py
led_builtin0.py
led_builtin1.py
led_builtin2.py
led_builtin3.py
led_toggle.py
logdata_ex.py
potentiometer.py
Raspberry Pi Pico
lib
data.txt
Datalogging.py
main.py
tmp36data.txt
write_file.py
[data.txt]
1 filename = ("data.txt")
2 file = open(filename, "w")
3
4 data = "Hello World"
5 file.write(data)
6
7 file.close()
Shell
>>> %Run -c $EDITOR_CONTENT
>>>
write_file.py
[data.txt]
1 Hello World
MicroPython (Raspberry Pi Pico) • COM6

Save Data to File in a While Loop

```
from time import sleep

filename = "data.txt"
file = open(filename, "w")

while True:
    data = "Hello World\n"
    file.write(data)
    file.flush()
    sleep(5)

file.close()
```

We use **\n** for adding a New Line in each iteration.

Here, it is important that you use **flush()** inside the While loop in order to save ("flush") data to the file in each iteration. If not, the data may not be saved to the file if you suddenly unplug the power supply from the Raspberry Pi Pico, etc.

Thonny - C:\Users\hansha\OneDrive\Documents\Industrial IT and Automation\IoT\Raspberry Pi Pico\Code Examples\write_file_loop.py @ 1 : 23

File Edit View Run Tools Help

Files

This computer
C: \ Users \ hansha \ OneDrive \ Documents \ Industrial IT and Automation \ IoT \ Raspberry Pi Pico \ Code Examples

- button_pulldown.py
- button_pulldown_led.py
- button_pulldown_led_picozero.py
- button_pullup.py
- button_pullup2.py
- Datalogging.py
- i2c_scan.py
- i2c_tc74.py
- i2c_tc74v2.py
- i2c_tc74_dataanalysis.py
- i2c_tc74_datalogging.py
- led.py
- led2.py
- led_builtin0.py
- led_builtin1.py
- led_builtin2.py
- led_builtin3.py
- led_toggle.py
- logdata_ex.py

Raspberry Pi Pico

- lib
- data.txt
- Datalogging.py
- main.py
- tmp36data.txt

write_file_loop.py | [data.txt]

```
1  from time import sleep
2
3  filename = ("data.txt")
4  file = open(filename, "w")
5
6  while True:
7      data = "Hello World\n"
8      file.write(data)
9      file.flush()
10     sleep(5)
11
12 file.close()
```

Shell

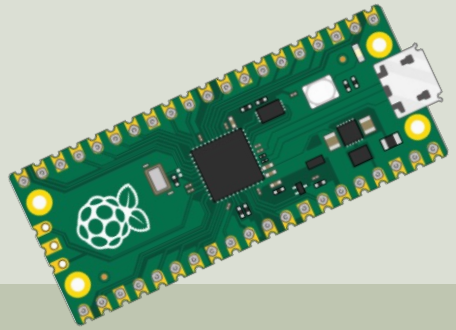
```
>>> %Run -c $EDITOR_CONTENT

  Traceback (most recent call last):
    File "<stdin>", line 10, in <module>
  KeyboardInterrupt:
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

write_file_loop.py | [data.txt]

```
1  Hello World
2  Hello World
3  Hello World
4  Hello World
5  Hello World
6  Hello World
7  Hello World
8
```

MicroPython (Raspberry Pi Pico) • COM6

Datalogging Example



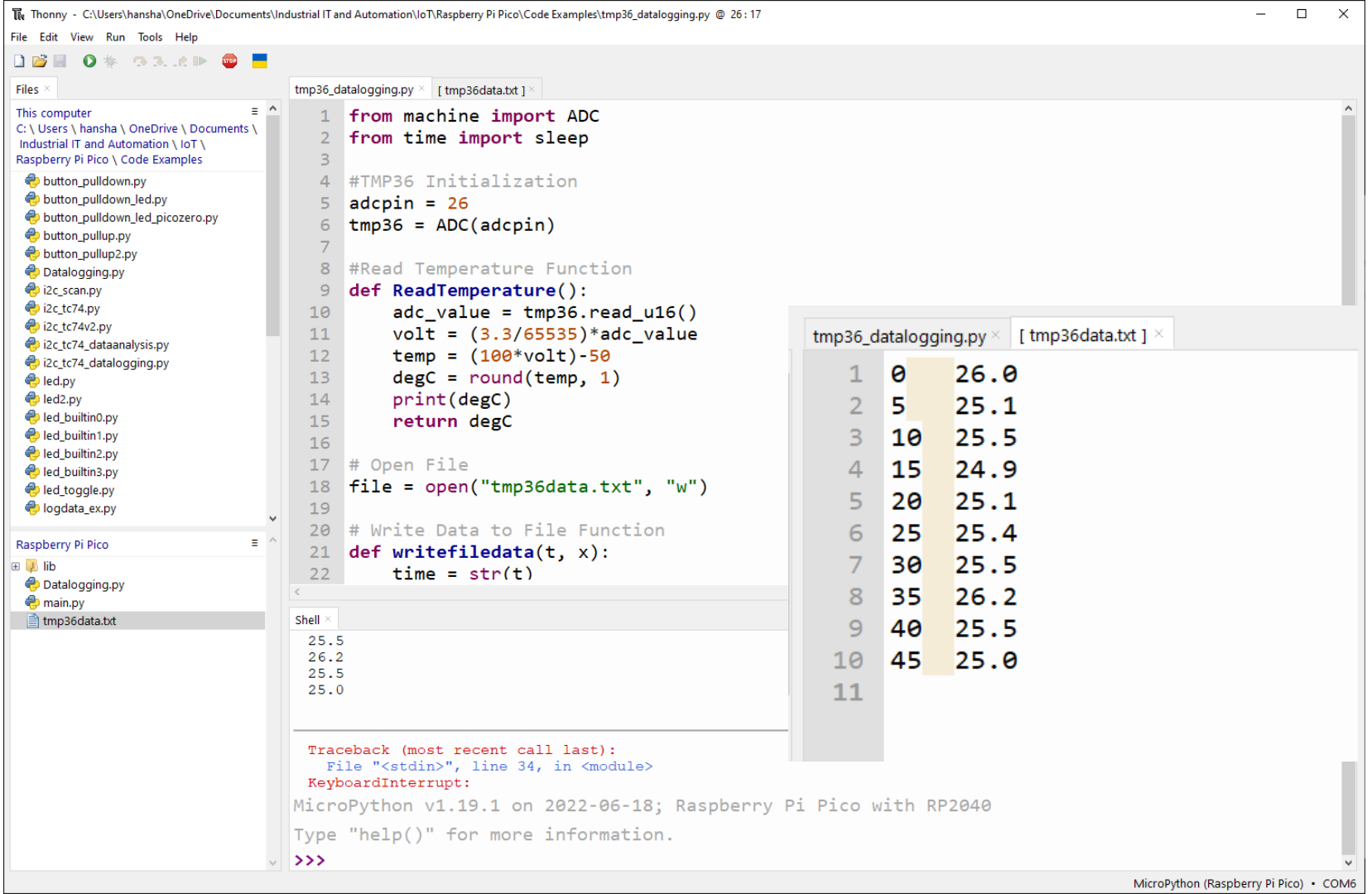
Hans-Petter Halvorsen

Table of Contents

Datalogging Example

- We will read data from a Temperature Sensor using Raspberry Pi Pico and MicroPython
- We will then Log Temperature Data on a File on the Pico Device

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

# Open File
file = open("tmp36data.txt", "w")

def writefiledata(t, x):
    time = str(t)
    value = str(round(x, 2))
    file.write(time + "\t" + value)
    file.write("\n")
    file.flush()

k = 0
Ts = 5
while True:
    degC = ReadTemperature()
    writefiledata(k*Ts, degC)
    k = k + 1
    sleep(Ts)
```


Thonny - C:\Users\hansha\OneDrive\Documents\Industrial IT and Automation\IoT\Raspberry Pi Pico\Code Examples\tmp36_datalogging.py @ 26 : 17
File Edit View Run Tools Help
Files
This computer
C: \ Users \ hansha \ OneDrive \ Documents \ Industrial IT and Automation \ IoT \ Raspberry Pi Pico \ Code Examples
button_pulldown.py
button_pulldown_led.py
button_pulldown_led_picozero.py
button_pullup.py
button_pullup2.py
Datalogging.py
i2c_scan.py
i2c_tc74.py
i2c_tc74v2.py
i2c_tc74_dataanalysis.py
i2c_tc74_datalogging.py
led.py
led2.py
led_builtin0.py
led_builtin1.py
led_builtin2.py
led_builtin3.py
led_toggle.py
logdata_ex.py
Raspberry Pi Pico
lib
Datalogging.py
main.py
tmp36data.txt
tmp36_datalogging.py
[tmp36data.txt]
1 from machine import ADC
2 from time import sleep
3
4 #TMP36 Initialization
5 adcpin = 26
6 tmp36 = ADC(adcpin)
7
8 #Read Temperature Function
9 def ReadTemperature():
10 adc_value = tmp36.read_u16()
11 volt = (3.3/65535)*adc_value
12 temp = (100*volt)-50
13 degC = round(temp, 1)
14 print(degC)
15 return degC
16
17 # Open File
18 file = open("tmp36data.txt", "w")
19
20 # Write Data to File Function
21 def writefiledata(t, x):
22 time = str(t)
tmp36_datalogging.py
[tmp36data.txt]
1 0 26.0
2 5 25.1
3 10 25.5
4 15 24.9
5 20 25.1
6 25 25.4
7 30 25.5
8 35 26.2
9 40 25.5
10 45 25.0
11
Shell
25.5
26.2
25.5
25.0
Traceback (most recent call last):
File "<stdin>", line 34, 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

Improved Code v2

- We will create a Separate Python Module (a separate Python File) with 2 Classes
- Class Sensor deals with the sensor reading
- Class File deals with the file writing

Datalogging.py

```
from machine import ADC

class Sensor:
    def __init__(self, pin):
        self.sensor = ADC(pin)

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

class File:
    def __init__(self, filename):
        self.file = open(filename, "w")

    def WriteData(self, t, x):
        time = str(t)
        value = str(round(x, 2))
        self.file.write(time + "\t" + value)
        self.file.write("\n")
        self.file.flush()
```

Logdata_ex.py

```
from Datalogging import Sensor, File
from time import sleep

adcpin = 26
tmp36 = Sensor(adcpin)

filename = "tmp36data.txt"
myfile = File(filename)

k = 0
Ts = 5
while True:
    degC = tmp36.ReadTemperature()
    myfile.WriteData(k*Ts, degC)
    k = k + 1
    sleep(Ts)
```

Improved Code v3

- We want to run the Datalogging without have a PC attached to the Pico
- We need to save the code as “**main.py**”, then this code will run when we plug the Pico to a Power Supply (PS)
- Finally, since we don’t see if the code is running or not on the Pico, I have added a code update that toggles the built-in LED in each iteration inside the While loop

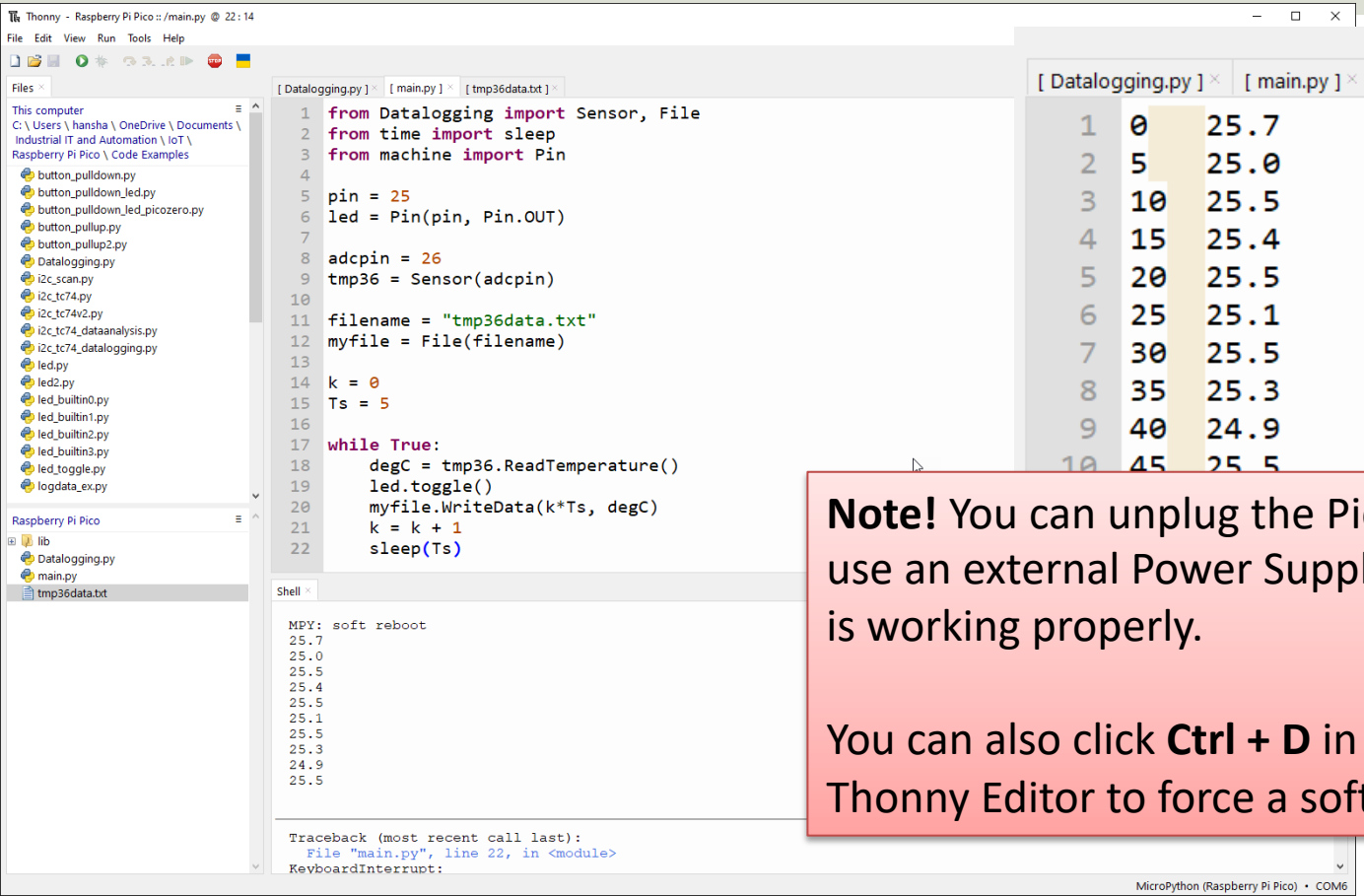
Datalogging.py

```
from machine import ADC

class Sensor:
    def __init__(self, pin):
        self.sensor = ADC(pin)

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

class File:
    def __init__(self, filename):
        self.file = open(filename, "w")

    def WriteData(self, t, x):
        time = str(t)
        value = str(round(x, 2))
        self.file.write(time + "\t" + value)
        self.file.write("\n")
        self.file.flush()
```

main.py

```
from Datalogging import Sensor, File
from time import sleep
from machine import Pin

pin = 25
led = Pin(pin, Pin.OUT)

adcpin = 26
tmp36 = Sensor(adcpin)

filename = "tmp36data.txt"
myfile = File(filename)

k = 0
Ts = 5

while True:
    degC = tmp36.ReadTemperature()
    led.toggle()
    myfile.WriteData(k*Ts, degC)
    k = k + 1
    sleep(Ts)
```

Results



Note! You can unplug the Pico from your PC and use an external Power Supply to see if the program is working properly.

You can also click **Ctrl + D** in the Shell inside the Thonny Editor to force a soft reboot command

Data Analysis Example

Python

Hans-Petter Halvorsen

Table of Contents

Data Analysis Example

- 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

```
import matplotlib.pyplot as plt

# Open File
f = open("tmp36data.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(float(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()
```

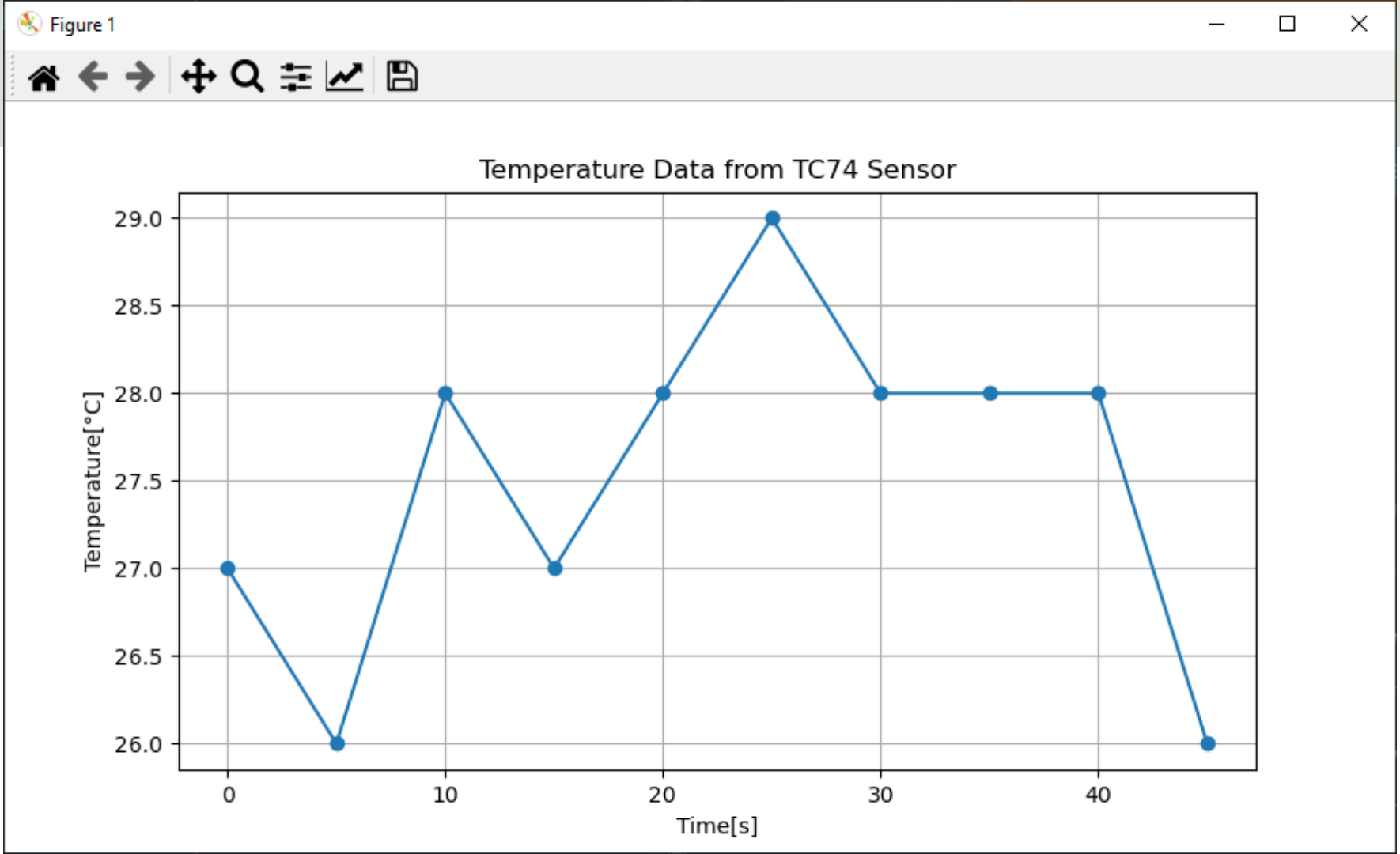


Figure 1
Temperature Data from TC74 Sensor
29.0
28.5
28.0
27.5
27.0
26.5
26.0
Temperature[°C]
0
10
20
30
40
Time[s]

Final Datalogging and Analysis Solution

Hans-Petter Halvorsen

Table of Contents

Datalogging

Datalogging.py

```
from machine import ADC
import time

class Sensor:
    def __init__(self, pin):
        self.sensor = ADC(pin)

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

class File:
    def __init__(self, filename):
        self.file = open(filename, "w")
        self.file.write("TimeStamp" + "\t" + "TemperatureValue" + "\n")

    def WriteData(self, t, x):
        time = str(t)
        value = str(round(x, 2))
        self.file.write(time + "\t" + value)
        self.file.write("\n")
        self.file.flush()

    def GetDateTime(self):
        datetime = time.localtime()

        year = str(datetime[0])

        month = str(datetime[1])
        if (len(month) == 1):
            month = "0" + month

        day = str(datetime[2])
        if (len(day) == 1):
            day = "0" + day

        hour = str(datetime[3])
        if (len(hour) == 1):
            hour = "0" + hour

        minute = str(datetime[4])
        if (len(minute) == 1):
            minute = "0" + minute

        second = str(datetime[5])
        if (len(second) == 1):
            second = "0" + second

        d = year + "." + month + "." + day
        t = hour + ":" + minute + ":" + second
        timestamp = d + " " + t
        return timestamp
```

main.py

```
from Datalogging import Sensor, File
from time import sleep
from machine import Pin

pin = 25
led = Pin(pin, Pin.OUT)

adcpin = 26
tmp36 = Sensor(adcpin)

filename = "tmp36data.txt"
myfile = File(filename)

k = 0
Ts = 5

while True:
    led.on()
    degC = tmp36.ReadTemperature()
    timestamp = myfile.GetDateTime()
    print(" T = " + str(degC) + "°C" + " @ " + timestamp)

    myfile.WriteData(timestamp, degC)
    k = k + 1
    led.off()
    sleep(Ts)
```

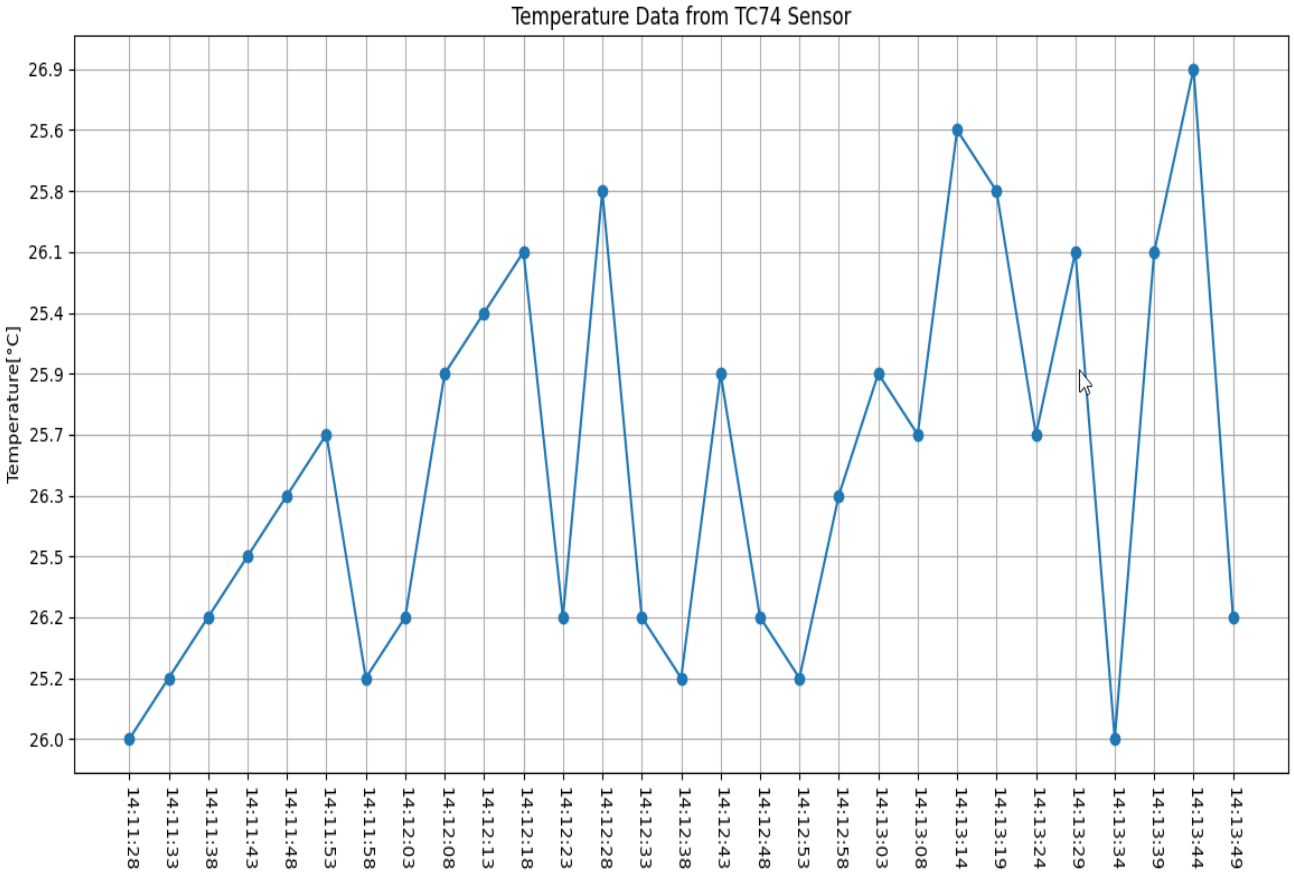
Data Analysis

```
import csv
import matplotlib.pyplot as plt

# Transform File Data into x Array and y Array that can be used for plotting
x = []
y = []
k = 1

log_file = open("tmp36data.txt", "r", encoding="utf8")
reader = csv.DictReader(log_file, delimiter="\t")
for record in reader:
    ts = record["TimeStamp"]
    ts = ts.split(" ")
    d = ts[0] #Datepart
    t = ts[1] #Timepart
    x.append(t)

    tv = record["TemperatureValue"]
    y.append(tv)

    k=k+1

plt.plot(x,y, '-o')
plt.title('Temperature Data from TC74 Sensor')
plt.xlabel('Time[s]')
plt.xticks(rotation=270)
plt.ylabel('Temperature[°C]')
plt.grid()
plt.show()
```

Results

[Datalogging.py] [main.py] dataanalysis.py tmp36data.txt

```
TimeStamp    TemperatureValue
2023.02.17 14:11:28 26.0
2023.02.17 14:11:33 25.2
2023.02.17 14:11:38 26.2
2023.02.17 14:11:43 25.5
2023.02.17 14:11:48 26.3
2023.02.17 14:11:53 25.7
2023.02.17 14:11:58 25.2
2023.02.17 14:12:03 26.2
2023.02.17 14:12:08 25.9
2023.02.17 14:12:13 25.4
2023.02.17 14:12:18 26.1
2023.02.17 14:12:23 26.2
2023.02.17 14:12:28 25.8
2023.02.17 14:12:33 26.2
2023.02.17 14:12:38 25.2
2023.02.17 14:12:43 25.9
2023.02.17 14:12:48 26.2
2023.02.17 14:12:53 25.2
2023.02.17 14:12:58 26.3
2023.02.17 14:13:03 25.9
2023.02.17 14:13:08 25.7
2023.02.17 14:13:14 25.6
2023.02.17 14:13:19 25.8
2023.02.17 14:13:24 25.7
2023.02.17 14:13:29 26.1
2023.02.17 14:13:34 26.0
2023.02.17 14:13:39 26.1
2023.02.17 14:13:44 26.9
```



Summary

- We have made a basic Datalogging application that can run on the Pico without having a PC attached to it
- The Data was stored on a local File on the Pico itself
- Then we copied the File to the PC and was doing some basic Data Analysis on the Data stored on the File
- The Datalogging and Data Analysis System was made in iterations until we get satisfying results

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