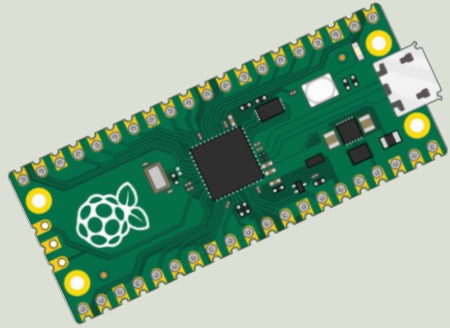


<https://www.halvorsen.blog>



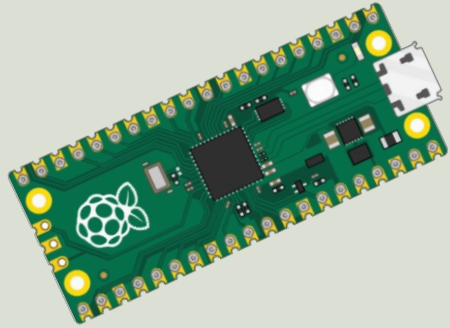
Raspberry Pi Pico

ThingSpeak IoT Cloud Service

Hans-Petter Halvorsen

Contents

- [Introduction](#)
- [Raspberry Pi Pico](#)
- [ThingSpeak](#)
- [MicroPython Examples](#)
 - [Read from built-in Temperature Sensor](#)
 - [Connect Raspberry Pi Pico W to WiFi](#)
 - [Write Data to ThingSpeak](#)
 - [Write to Multiple Fields in a ThingSpeak Channel](#)



Introduction

ThingSpeak IoT Cloud Service

Hans-Petter Halvorsen

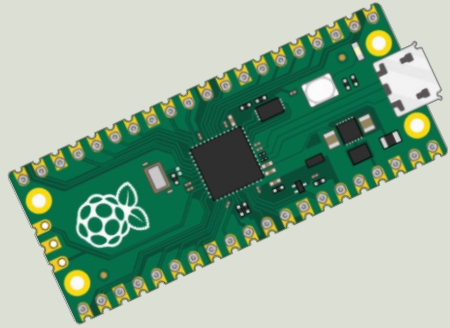
[Table of Contents](#)

Introduction

- In this Tutorial we will use **Raspberry Pi Pico W**
- We will write Data to a Cloud Service called **ThingSpeak**
- ThingSpeak is owned by MathWorks, the same vendor that develop the MATLAB software
- <https://thingspeak.com>
- We will use the built-in Temperature Sensor on the Pico hardware as an example when Writing Data to ThingSpeak (but you can use any type of sensor)

What do you need?

- **Raspberry Pi Pico W or WH**
- A Micro-USB cable
- A PC with Thonny Python Editor (or another Python Editor)
- Breadboard, Electronics Components like LED, Resistors, Jumper wires, etc.
- Sensors, e.g., TMP36 Temperature Sensor (In this Tutorial we will use the built in Temperature Sensor)



Raspberry Pi Pico

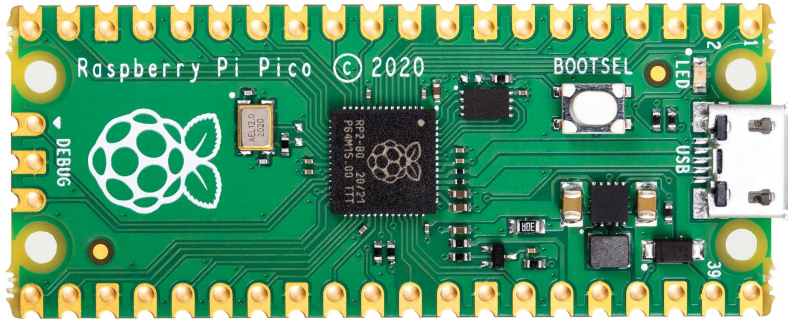
Raspberry Pi Pico

We have 4 different types:

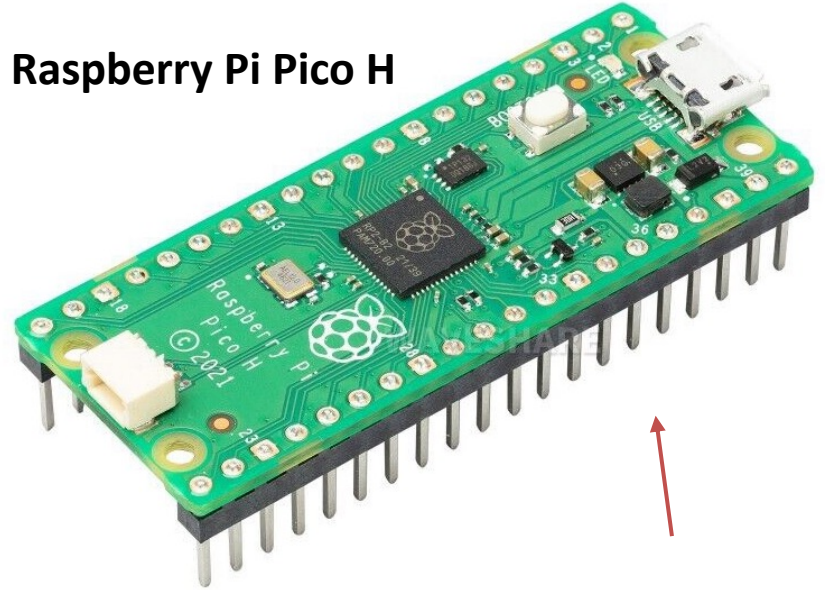
- Raspberry Pi Pico (original)
- Raspberry Pi Pico H - pre-soldered header pins included
- Raspberry Pi Pico W – WiFi included
- Raspberry Pi Pico WH – WiFi and pre-soldered header pins included

Raspberry Pi Pico Series

1. Raspberry Pi Pico (original)

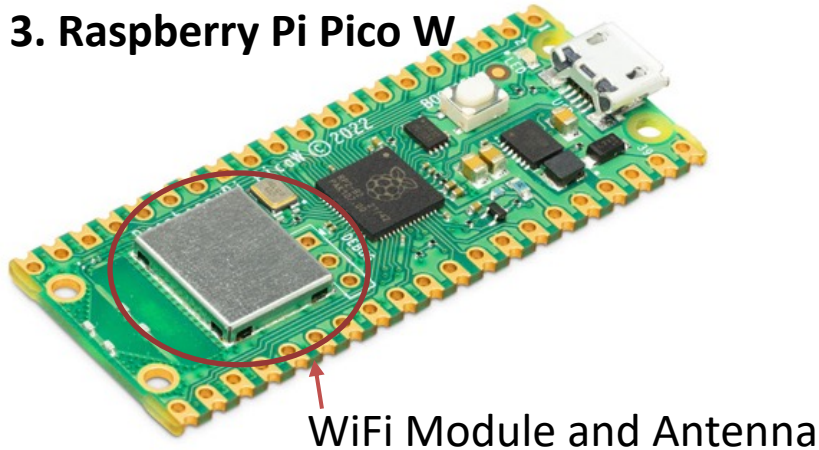


2. Raspberry Pi Pico H



Pre-soldered header pins included

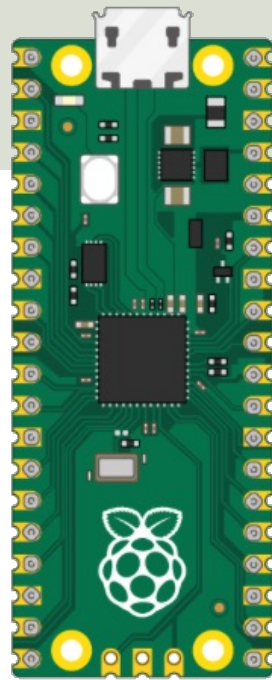
3. Raspberry Pi Pico W



4. Raspberry Pi Pico WH has WiFi and pre-soldered header pins included (no image here)

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/>

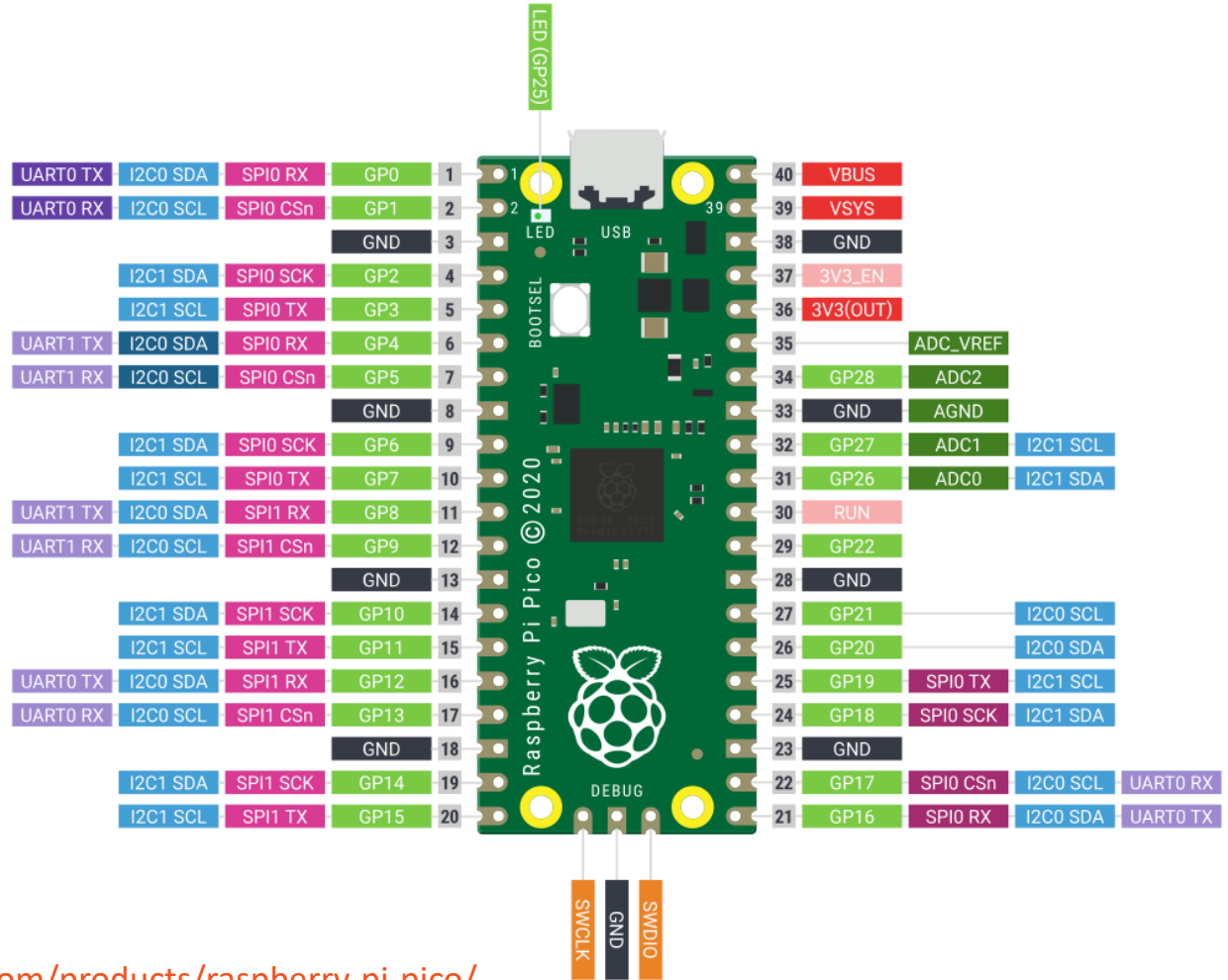
<https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico>

Raspberry Pi Pico W

- Raspberry Pi Pico W or WH has built-in support for WiFi and Wireless Communication
- Raspberry Pi Pico W offers 2.4GHz 802.11 b/g/n wireless WiFi support
- Getting started with your Raspberry Pi Pico W:
<https://projects.raspberrypi.org/en/projects/get-started-pico-w>
- Raspberry Pi Pico W Datasheet:
<https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf>

Pico Pinout

■	Power
■	Ground
■	UART / UART (default)
■	GPIO, PIO, and PWM
■	ADC
■	SPI / SPI (default)
■	I2C / I2C (default)
■	System Control
■	Debugging



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

- **Download MicroPython Firmware:**
<https://rpf.io/pico-w-firmware>
- Hold down the BOOTSEL button on your Raspberry Pi Pico W
- Connect the Pico to your PC using the USB cable
- Drag and drop the firmware file you downloaded to your Pico

Thonny

```
Thonny - C:\Temp\Raspberry Pi Pico\LED Example.py @ 3:1
File Edit View Run Tools Help
Files
This computer
C:\Temp\Raspberry Pi Pico
LED Example.py
PicoSensor.py
ReadTemp.py
Raspberry Pi Pico
TemperatureSensor.py
thermistor_ex2.py
LED Example.py
1 import machine
2 import time
3
4 pin = 16
5 led = machine.Pin(pin, machine.PIN_CONFIG_OUTPUT)
6
7 while True:
8     led.value(1)
9     time.sleep(2)
10    led.value(0)
11    time.sleep(2)
Shell
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
- Built-in support for the Raspberry Pi Pico hardware/MicroPython firmware
- Its free
- Download: <https://thonny.org>

<https://www.halvorsen.blog>

 ThingSpeak™



ThingSpeak

Hans-Petter Halvorsen

[Table of Contents](#)

ThingSpeak

- **ThingSpeak is an IoT service that lets you collect and store sensor data in the cloud and develop Internet of Things applications.**
- ThingSpeak is free for small non-commercial projects
- In addition, they offer different types of licenses where you pay a monthly fee
- ThingSpeak is owned by MathWorks, the same vendor that develop the MATLAB software
- <https://thingspeak.com>

ThingSpeak

Here you see an example of how Data can be presented in the ThingSpeak Web page

<https://thingspeak.com>

The screenshot displays the ThingSpeak web interface for a channel named "temperature". The top navigation bar includes "Channels", "Apps", "Devices", and "Support". The channel information shows "Channel ID: [redacted]", "Author: [redacted]", and "Access: Public". Navigation tabs include "Private View", "Public View", "Channel Settings", "Sharing", "API Keys", and "Data Import / Export". Action buttons for "Add Visualizations", "Add Widgets", "Export recent data", and "More Information" are visible, along with "MATLAB Analysis" and "MATLAB Visualization" buttons.

Channel Stats
Created: 4 years ago
Last entry: less than a minute ago
Entries: 242

Field 1 Chart: Office Temperature
A line chart showing "Office Temperature [C]" on the y-axis (ranging from 20 to 22) against "Date" on the x-axis (ranging from 15:00 to 15:10). The data shows a regular oscillating pattern between approximately 19.5°C and 22.5°C.

Field 2 Chart: Outdoor Temperature
A line chart showing "Outdoor Temperature [C]" on the y-axis against "Date" on the x-axis. The chart area is currently blank, indicating no data is displayed.

Field 3 Chart: TMP36 Temperature
A line chart showing "TMP36 Temperature" on the y-axis (ranging from 28 to 29) against "Date" on the x-axis (ranging from 11:45 to 12:30). The data shows a fluctuating pattern between approximately 27.5 and 29.5.

Field 4 Chart: Work
A line chart showing "Work" on the y-axis against "Date" on the x-axis. The chart area is currently blank, indicating no data is displayed.

ThingSpeak

- It works with Arduino, Raspberry Pi and MATLAB (premade libraries and APIs exists).
- But it should work with all kind of Programming Languages, since it uses a **REST API** and **HTTP**.

ThingSpeak – Channel Settings

Channel ID: [redacted]
Author: hansha
Access: Public

temperature

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Channel Settings

Percentage complete 65%

Channel ID [redacted]

Name Work

Description

Field 1 Office Temperature [C]

Field 2 Temperature B Buildin

Field 3 Tout

Field 4 Kp

Field 5 Ti

Field 6 SP

Field 7 Field7

Field 8 Field8

Help

Channels store all the data that a ThingSpeak app collects. Each channel can have up to 8 fields. Fields are the data points that you collect, plus the status data. Once you collect data in a channel, you can visualize it.

Channel Settings

- **Percentage complete:** Calculated based on the number of fields that are complete. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
 - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
 - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
 - **Elevation:** Specify the elevation position meters. For example, the elevation of the city of London is 35.052.

You can set up different Channels in ThingSpeak.

Each Channel can have up to 8 Fields.

You can have up to 4 different Channels for the Free License.

ThingSpeak - REST API

The screenshot shows the ThingSpeak API management interface. At the top, it displays channel information: Channel ID, Author, and Access (Public). Below this are navigation tabs: Private View, Public View, Channel Settings, Sharing, API Keys, and Data. The 'API Keys' tab is selected, showing two sections: 'Write API Key' and 'Read API Keys'. The 'Write API Key' section has a 'Key' input field and a 'Generate New Write API Key' button. The 'Read API Keys' section has a 'Key' input field, a 'Note' text area, and 'Save Note' and 'Delete API Key' buttons. On the right, there is a 'Help' section with 'API Keys Settings' and 'API Requests'.

Channel ID: [redacted] | temperature

Author: [redacted]

Access: Public

Private View Public View Channel Settings Sharing API Keys Data

Write API Key

Key [redacted]

Generate New Write API Key

Read API Keys

Key [redacted]

Note [redacted]

Save Note Delete API Key

Add New Read API Key

Key needed to Write Data to the Channel

Key needed to Read Data from the Channel

Help

API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=[redacted]&field=[redacted]
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/[redacted]/feeds.json?results=2
```

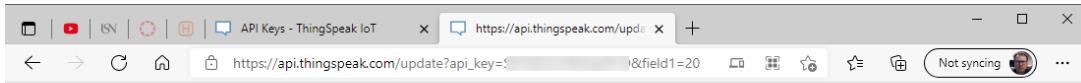
Read a Channel Field

```
GET https://api.thingspeak.com/channels/[redacted]/fields/1.json?results=
```

REST API – Write Data

Use your standard Web Browser (e.g., Microsoft Edge, or Google Chrome) and enter the following:

`https://api.thingspeak.com/update?api_key=XXXXXXXXXXXXXXXXXXXX&field1=20`



Your **Write API Key**

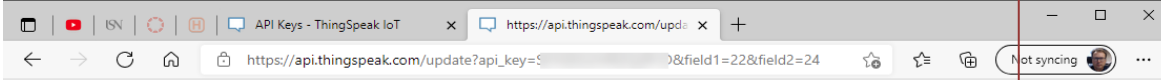
Field Number 1-8

Value

REST API – Write Multiple Fields

Use your standard Web Browser (e.g., Microsoft Edge, or Google Chrome) and enter the following:

`https://api.thingspeak.com/update?api_key=XXXXXXXXXXXXXXXXXXXX&field1=21&field2=24`



21

Your **Write API Key**

Field + Value

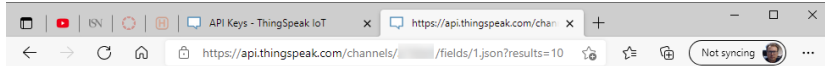
Field + Value Etc.

REST API – Read Data

Use your standard Web Browser (e.g., Microsoft Edge, or Google Chrome) and enter the following:

Data Format (JSON or XML)

`https://api.thingspeak.com/channels/xxxxxx/fields/1.json?results=10`



```
{ "channel": { "id": "xxxxxx", "name": "Work", "latitude": "0.0", "longitude": "0.0", "field1": "Office Temperature [C]", "field2": "Temperature B  
Building [C]", "field3": "Tou", "field4": "Kp", "field5": "T1", "field6": "Sp", "field7": "Field7", "field8": "field8", "created_at": "2017-05-  
30T11:41:00Z", "updated_at": "2021-09-09T10:59:27Z", "last_entry_id": 21, "feeds": [{"created_at": "2021-09-  
08T12:54:04Z", "entry_id": 12, "field1": null}, {"created_at": "2021-09-08T13:03:54Z", "entry_id": 13, "field1": null}, {"created_at": "2021-09-  
09T09:27:34Z", "entry_id": 14, "field1": "20.00"}, {"created_at": "2021-09-09T09:34:38Z", "entry_id": 15, "field1": null}, {"created_at": "2021-  
09-09T09:35:35Z", "entry_id": 16, "field1": "18.00"}, {"created_at": "2021-09-09T10:46:11Z", "entry_id": 17, "field1": "0.00"},  
{"created_at": "2021-09-09T10:48:45Z", "entry_id": 18, "field1": "21"}, {"created_at": "2021-09-09T11:06:32Z", "entry_id": 19, "field1": "20"},  
{"created_at": "2021-09-09T11:09:46Z", "entry_id": 20, "field1": "21"}, {"created_at": "2021-09-09T11:17:08Z", "entry_id": 21, "field1": "22"}]}
```

Your Channel ID

Field Number

Resulting JSON String with Data

Number of Data Points, e.g., 1 for only the last value, 10 for the last 10 values, etc.

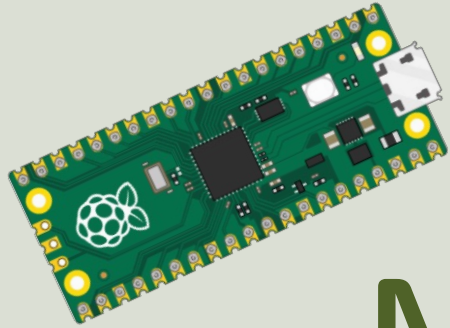
REST API – Read Data (JSON)

<https://api.thingspeak.com/channels/xxxxxx/fields/1.json?results=10>

```
{"channel":{"id":"xxxxxx","name":"Work","latitude":"0.0","longitude":"0.0","field1":"Office Temperature [C]","field2":"Temperature B Building [C]","field3":"Tout","field4":"Kp","field5":"Ti","field6":"SP","field7":"Field7","field8":"Field8","created_at":"2017-05-30T11:41:00Z","updated_at":"2021-09-09T10:59:27Z","last_entry_id":21},  
"feeds":[  
  {"created_at":"2021-09-08T12:54:04Z","entry_id":12,"field 1":null},  
  {"created_at":"2021-09-08T13:03:54Z","entry_id":13,"field 1":null},  
  {"created_at":"2021-09-09T09:27:34Z","entry_id":14,"field 1":"20.00"},  
  {"created_at":"2021-09-09T09:34:38Z","entry_id":15,"field 1":null},  
  {"created_at":"2021-09-09T09:35:35Z","entry_id":16,"field 1":"18.00"},  
  {"created_at":"2021-09-09T10:46:11Z","entry_id":17,"field 1":"0.00"},  
  {"created_at":"2021-09-09T10:48:45Z","entry_id":18,"field 1":"25"},  
  {"created_at":"2021-09-09T11:06:32Z","entry_id":19,"field 1":"20"},  
  {"created_at":"2021-09-09T11:09:46Z","entry_id":20,"field 1":"21"},  
  {"created_at":"2021-09-09T11:17:08Z","entry_id":21,"field 1":"22"}  
]}
```

Values

We need to parse the JSON string in order to get the actual Values



MicroPython Examples

Hans-Petter Halvorsen

[Table of Contents](#)

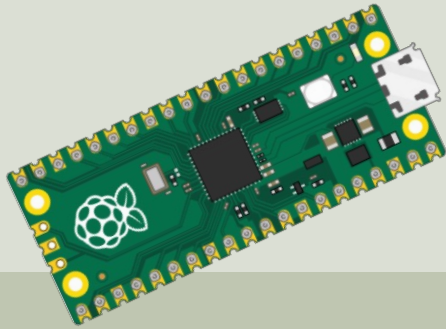
MicroPython Examples

The final goal is to write Data from a Sensor to the ThingSpeak Cloud Service. We take it step by step:

- Use the built-in Temperature Sensor
- Connect Raspberry Pi Pico W to WiFi
- Write Data to ThingSpeak
- Modify and improve until we get a final system that fulfills our requirements



Built-in Temperature Sensor



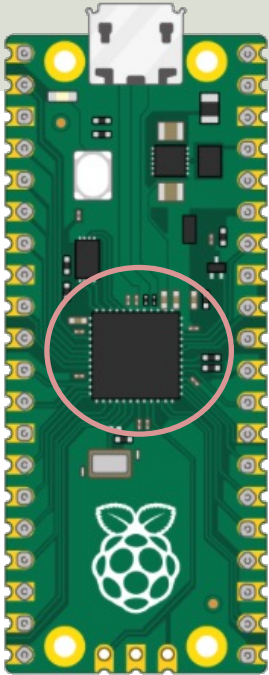
Hans-Petter Halvorsen

[Table of Contents](#)

Built-in Temperature Sensor

- The raspberry Pi Pico has a built-in Temperature Sensor
- The Temperature Sensor is inside the RP2040 microcontroller chip which is located on the Raspberry Pi Pico
- It can be used for simple application and for test and demo purposes

RP2040 Datasheet



4.9.5. Temperature Sensor

The temperature sensor measures the V_{be} voltage of a biased bipolar diode, connected to the fifth ADC channel (AINSEL=4). Typically, $V_{be} = 0.706V$ at 27 degrees C, with a slope of $-1.721mV$ per degree. Therefore the temperature can be approximated as follows:

$$T = 27 - (ADC_voltage - 0.706)/0.001721$$

As the V_{be} and the V_{be} slope can vary over the temperature range, and from device to device, some user calibration may be required if accurate measurements are required.

The temperature sensor's bias source must be enabled before use, via CS.TS_EN. This increases current consumption on ADC_AVDD by approximately $40\mu A$.

i NOTE

The on board temperature sensor is very sensitive to errors in the reference voltage. If the ADC returns a value of 891 this would correspond to a temperature of $20.1^{\circ}C$. However if the reference voltage is 1% lower than 3.3V then the same reading of 891 would correspond to $24.3^{\circ}C$. You would see a change in temperature of over $4^{\circ}C$ for a small 1% change in reference voltage. Therefore if you want to improve the accuracy of the internal temperature sensor it is worth considering adding an external reference voltage.

Datasheet (Chapter 4.9.5. Temperature Sensor)

<https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf>

Built-in Temperature Sensor

```
import machine
import time

adcpin = 4
sensor = machine.ADC(adcpin)

def ReadTemperature():
    adc_value = sensor.read_u16()
    volt = (3.3/65535)*adc_value
    temperature = 27 - (volt - 0.706)/0.001721
    return round(temperature, 1)

while True:
    temperature = ReadTemperature()
    print(temperature)
    time.sleep(5)
```



temperature_sensor_builtint.py ×

```
1 import machine
2 import time
3
4 adcpin = 4
5 sensor = machine.ADC(adcpin)
6
7 def ReadTemperature():
8     adc_value = sensor.read_u16()
9     volt = (3.3/65535)*adc_value
10    temperature = 27 - (volt - 0.706)/0.001721
11    return round(temperature, 1)
12
13
14 while True:
15     temperature = ReadTemperature()
16     print(temperature)
17     time.sleep(5)
```

Shell ×

>>> %Run -c \$EDITOR_CONTENT

```
27.0
27.5
27.0
26.6
26.1
24.2
24.2
```

```
from machine import ADC
```

```
class Temperature:
```

```
    def __init__(self):  
        adcpin = 4  
        self.sensor = ADC(adcpin)
```

```
    def ReadTemperature(self):
```

```
        adc_value = self.sensor.read_u16()  
        volt = (3.3/65535)*adc_value  
        temperature = 27 - (volt - 0.706)/0.001721  
        return round(temperature, 1)
```

Main Program:

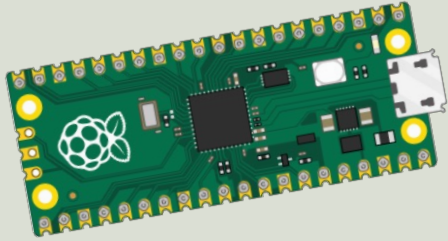
```
from PicoSensor import Temperature  
import time
```

```
sensor = Temperature()
```

```
while True:
```

```
    temperature = sensor.ReadTemperature()  
    print(temperature)  
    time.sleep(5)
```

You may also want to structure the code into a Class and a separate Python module to make it easier to reuse the code in different applications



Connect Raspberry Pico W to WiFi

Hans-Petter Halvorsen

[Table of Contents](#)

MicroPython **network** Module

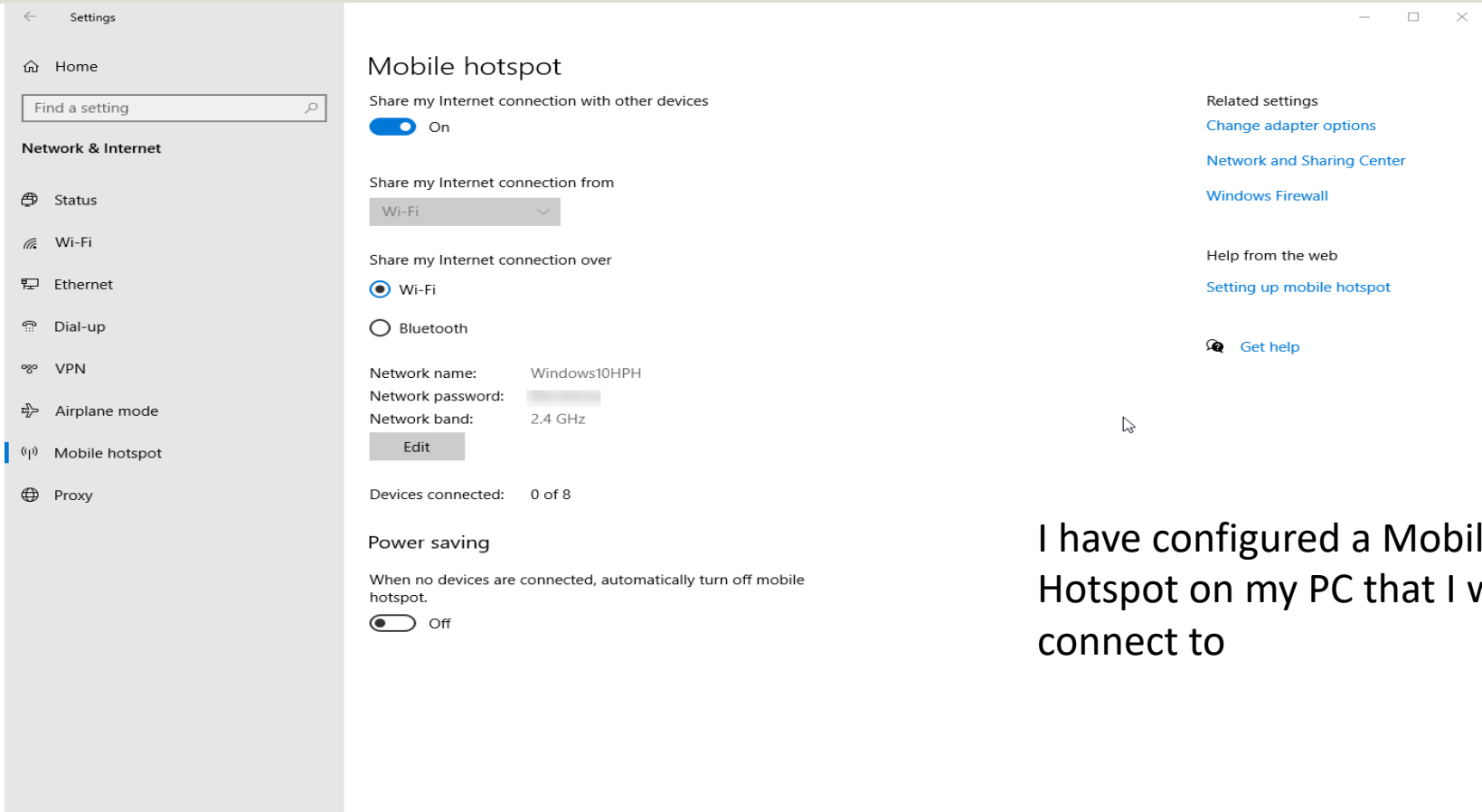
- In order to connect to network and WiFi and use network communication on Raspberry Pi Pico you need to use the MicroPython **network** Module
- Important Functions/Properties are:

```
wlan = network.WLAN(network.STA_IF)  
wlan.active(True)  
wlan.connect(ssid, password)  
wlan.isconnected()
```

<https://projects.raspberrypi.org/en/projects/get-started-pico-w>

<https://docs.micropython.org/en/latest/library/network.html#common-network-adapter-interface>

Mobile Hotspot



The image shows a Windows Settings window titled "Mobile hotspot". On the left is a navigation pane with "Settings" at the top, a "Home" button, a search box, and a list of categories: "Network & Internet", "Status", "Wi-Fi", "Ethernet", "Dial-up", "VPN", "Airplane mode", "Mobile hotspot" (highlighted with a blue bar), and "Proxy". The main content area is titled "Mobile hotspot" and contains the following sections:

- Share my Internet connection with other devices:** A toggle switch is turned "On".
- Share my Internet connection from:** A dropdown menu is set to "Wi-Fi".
- Share my Internet connection over:** Two radio buttons are shown: "Wi-Fi" (selected) and "Bluetooth".
- Network details:** Network name: "Windows10HPH", Network password: [blacked out], Network band: "2.4 GHz". Below this is an "Edit" button.
- Devices connected:** "0 of 8".
- Power saving:** A toggle switch is turned "Off". Below it is the text: "When no devices are connected, automatically turn off mobile hotspot."

On the right side of the main content area, there are links for "Related settings": "Change adapter options", "Network and Sharing Center", and "Windows Firewall". Below these are links for "Help from the web": "Setting up mobile hotspot" and "Get help".

I have configured a Mobile Hotspot on my PC that I will connect to

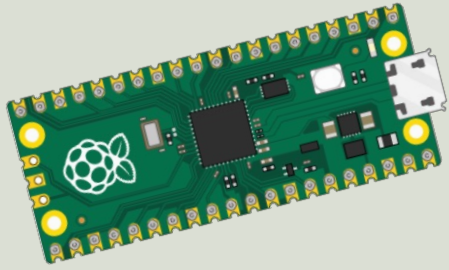
Connect to WiFi

```
import machine
import network
from time import sleep

ssid = "xxxxxxx"
password = "xxxxxxx"

def ConnectWiFi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    while wlan.isconnected() == False:
        print('Waiting for connection...')
        sleep(1)
    ip = wlan.ifconfig()[0]
    print(f'Connected on {ip}')
    return ip

try:
    ip = ConnectWiFi()
except KeyboardInterrupt:
    machine.reset()
```



Write to ThingSpeak

Raspberry Pi Pico W

Hans-Petter Halvorsen

[Table of Contents](#)

Write to ThingSpeak

- **Each channel has up to 8 data fields, location fields, and a status field.**
- **Note!** A Free ThingSpeak Channel can only be updated every **15** seconds
- We will use the **ThingSpeak REST API** and HTTP.

MicroPython **urequests** Module

- In order to use **HTTP** with Raspberry Pi Pico W we can use the **urequests** MicroPython Module
- It is included with the MicroPython firmware that you already has installed on your Raspberry Pi Pico (so no extra download/installation is necessary)
- **The urequests module allows you to send HTTP requests using MicroPython**
- The urequests module is a downscaled version of the requests module that exists for ordinary Python

https://www.w3schools.com/python/module_requests.asp

https://makeblock-micropython-api.readthedocs.io/en/latest/public_library/Third-party-libraries/urequests.html

MicroPython urequests Module

In order to send (post) Data you can write like this:

```
urequests.post(url)
```

Field number (1-8)

value

The ThingSpeak API is like this:

```
https://api.thingspeak.com/update?api_key=XXXXXX&field1=20
```



Then you can write like this:

```
url = "https://api.thingspeak.com/update?api_key=XXXXXX&field1=20"  
urequests.post(url)
```


ThingSpeak Write Example

```
import network
import urequests
import random
from time import sleep

#Network Initialization
ssid = "xxxxxxx"
password = "xxxxxxx"

def ConnectWiFi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    while wlan.isconnected() == False:
        print('Waiting for connection...')
        sleep(1)
    ip = wlan.ifconfig()[0]
    print(f'Connected on {ip}')
    return ip

#Connect to Network
ip = ConnectWiFi()

#ThingSpeak Initialization
server = "http://api.thingspeak.com/"
apikey = "xxxxxxx"
field = 1

#Main Program
while True:
    temperature = random.uniform(20, 25) #Random Number between 20 and 25
    temperature = round(temperature, 1)
    print(f"T = {temperature}°C")

    url = f"{server}/update?api_key={apikey}&field{field}={temperature}"
    request = urequests.post(url)
    request.close()

    sleep(20)
```

Code Improvements

We do some improvements in the code

- We use the Temperature Sensor instead
- Improve the code quality
- Etc.

```
from PicoSensor import Temperature
import machine
import network
import urequests
from time import sleep
from mywifi import GetMyWiFi

#Network Initialization
ssid, password = GetMyWiFi()

def ConnectWiFi():
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    while wlan.isconnected() == False:
        print('Waiting for connection...')
        sleep(1)
    ip = wlan.ifconfig()[0]
    print(f'Connected on {ip}')
    return ip

#Connect to Network
ip = ConnectWiFi()
```

Code Part 1

```
#Sensor Initialization
sensor = Temperature()

#ThingSpeak Initialization
server = "http://api.thingspeak.com/"
apikey = "IEQ6WFMNE99LNLIS"
field = 1

#Main Program
while True:
    temperature = sensor.ReadTemperature()
    print(f"T={temperature}°C")

    url = f"{server}/update?api_key={apikey}&field{field}={temperature}"
    print(url)

    request = urequests.post(url)
    print(request)

    sleep(20)
```

Code Part 2

Code Improvements

- We make separate Python Modules for Network part and for ThingSpeak part
- We also put Usernames, Passwords, etc. into separate Configuration Files
- Basically, most of the Code (that are general and can be reused) are put into separate Python Modules to improve the code structure and code quality and make it easier to reuse the code for different applications

WiFi

wificonfig.py

```
ssid = "xxx"  
password = "xxx"
```

WiFiNetwork.py

```
import network  
import wificonfig  
from time import sleep  
  
class WiFi:  
    def __init__(self):  
        self.ssid = wificonfig.ssid  
        self.password = wificonfig.password  
  
    def ConnectWiFi(self):  
        wlan = network.WLAN(network.STA_IF)  
        wlan.active(True)  
        wlan.connect(self.ssid, self.password)  
        while wlan.isconnected() == False:  
            print('Waiting for connection...')  
            sleep(1)  
        ip = wlan.ifconfig()[0]  
        print(f'Pico Connected on IP {ip}')  
        return ip
```

ThingSpeak

ThingSpeak.py

```
import urequests
import thingspeakconfig

class ThingSpeakApi:
    def __init__(self, field):
        self.server = thingspeakconfig.server
        self.apikey = thingspeakconfig.apikey
        self.field = field

    def WriteData(self, fieldvalue):
        url = f"{self.server}/update?api_key={self.apikey}&field{self.field}={fieldvalue}"
        #print(url)

        request = urequests.post(url)
        request.close()
        #print(request)
```

Make sure to include request.close()

thingspeakconfig.py

```
server = "http://api.thingspeak.com/"
apikey = "xxxxxx"
```

Main Program



Most of the Code are put into separate Python libraries to improve the code structure and code quality and make it easier to reuse the code for different applications

```
from PicoSensor import Temperature
from WiFiNetwork import WiFi
from ThingSpeak import ThingSpeakApi
from time import sleep

#Sensor Initialization
sensor = Temperature()

#ThingSpeak Initialization
field = 1
thingspeak = ThingSpeakApi(field)

#Network Initialization
network = WiFi()
ip = network.ConnectWiFi()

#Main Program
while True:
    temperature = sensor.ReadTemperature()
    print(f"T={temperature}°C")

    thingspeak.WriteData(temperature)

    sleep(20)
```



Files

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

Raspberry Pi Pico

lib
mywifi.py
PicoSensor.py
ThingSpeak.py
Thingspeak2.py
thingspeakconfig.py
wificonfig.py
WiFiNetwork.py

thingspeak_write2.py

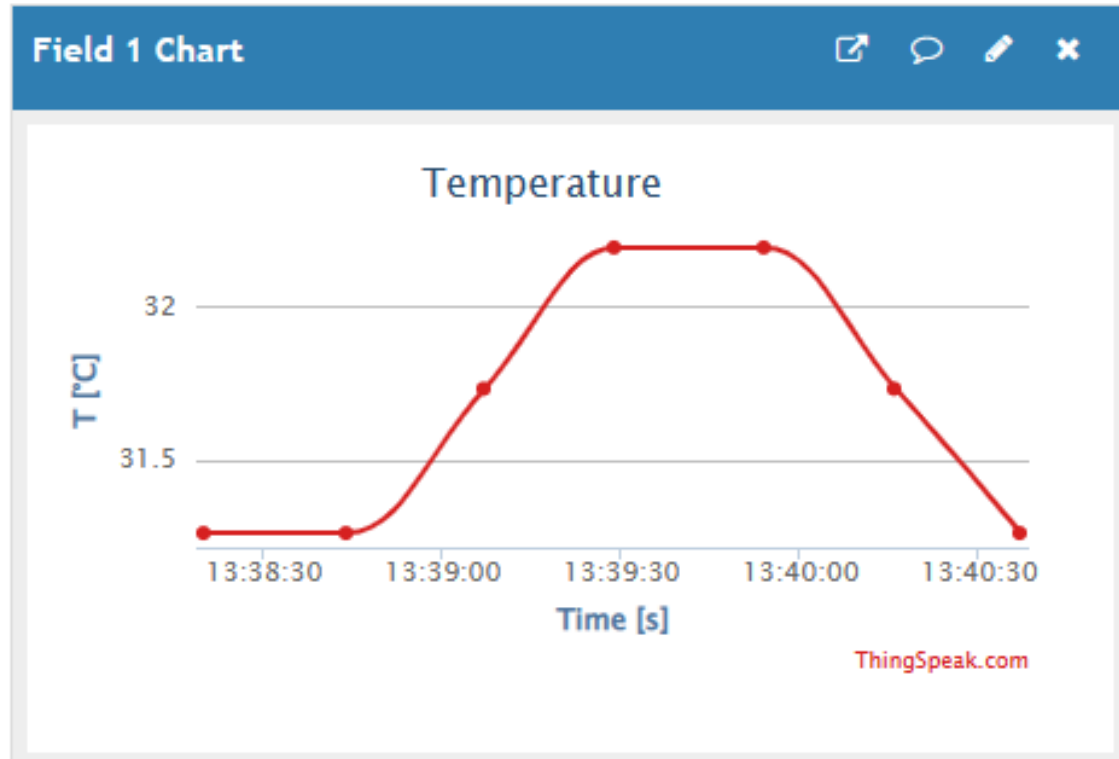
```
1 from PicoSensor import Temperature
2 from WiFiNetwork import WiFi
3 from ThingSpeak import ThingSpeakApi
4 from time import sleep
5
6 #Sensor Initialization
7 sensor = Temperature()
8
9 #ThingSpeak Initialization
10 field = 1
11 thingspeak = ThingSpeakApi(field)
12
13 #Network Initialization
14 network = WiFi()
15 ip = network.ConnectWiFi()
16
17 #Main Program
18 while True:
19     temperature = sensor.ReadTemperature()
20     print(f"T={temperature}°C")
21
22     thingspeak.WriteData(temperature)
23
24     sleep(20)
```

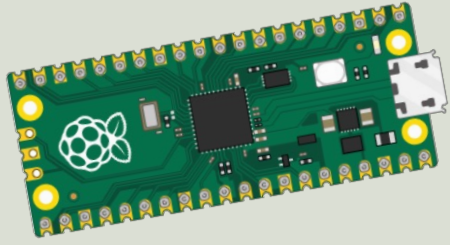
Shell

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

```
Pico Connected on IP 192.168.137.68
T=31.26°C
T=31.26°C
T=31.73°C
T=32.19°C
T=32.19°C
T=31.73°C
T=31.26°C
```


ThingSpeak Website





Write to Multiple Fields

Write Multiple Fields

In a ThingSpeak Channel we can use up to **8 Fields**

This is an example of the ThingSpeak API for writing data to multiple fields:

```
https://api.thingspeak.com/update?api_key=XXXXXX&field1=21&field2=24
```

Let's update/modify the code so it is possible to write data to multiple Fields in a Channel as well

ThingSpeak.py

```
import urequests
import thingspeakconfig

class ThingSpeakApi:
    def __init__(self):
        self.server = thingspeakconfig.server
        self.apikey = thingspeakconfig.apikey

    def WriteSingleField(self, fieldvalue):
        url = f"{self.server}/update?api_key={self.apikey}&field1={fieldvalue}"
        request = urequests.post(url)
        request.close()

    def WriteMultipleFields(self, field_data):
        url = f"{self.server}/update?api_key={self.apikey}"
        i = 1
        for field_value in field_data:
            url = url + f"&field{i}={field_value}"
            i = i + 1

        request = urequests.post(url)
        request.close()
```

Main Program

```
from PicoSensor import Temperature
from WiFiNetwork import WiFi
from ThingSpeak import ThingSpeakApi
from time import sleep
```

```
#Sensor Initialization
sensor = Temperature()
```

```
#ThingSpeak Initialization
thingspeak = ThingSpeakApi()
```

```
#Network Initialization
network = WiFi()
ip = network.ConnectWiFi()
```

```
#Main Program
while True:
    temperature = sensor.ReadTemperature()
    print(f"T = {temperature}°C")

    temperatureF = round((temperature*1.8) + 32, 2)

    field_data = (temperature, temperatureF)
    thingspeak.WriteMultipleFields(field_data)

    sleep(20)
```



Files

This computer

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

Raspberry Pi Pico

lib

mywifi.py
PicoSensor.py
ThingSpeak.py
ThingSpeak2.py
thingspeakconfig.py
wificonfig.py
WiFiNetwork.py

[ThingSpeak2.py] < thingspeak_write3.py >

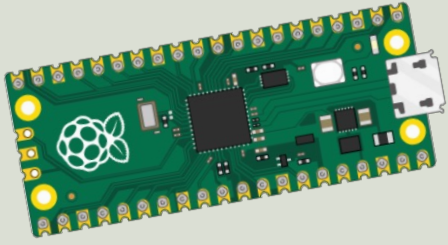
```
1 from PicoSensor import Temperature
2 from WiFiNetwork import WiFi
3 from ThingSpeak2 import ThingSpeakApi
4 from time import sleep
5
6 #Sensor Initialization
7 sensor = Temperature()
8
9 #ThingSpeak Initialization
10 field = 1
11 thingspeak = ThingSpeakApi(field)
12
13 #Network Initialization
14 network = WiFi()
15 ip = network.ConnectWiFi()
16
17 #Main Program
18 while True:
19     temperature = sensor.ReadTemperature()
20     print(f"T = {temperature}°C")
21
22     temperatureF = round((temperature*1.8) + 32, 2)
23
24     field_data = (temperature, temperatureF)
25     thingspeak.WriteMultipleFields(field_data)
26
27     sleep(20)
```

Shell

>>> %Run -c \$EDITOR_CONTENT

```
Pico Connected on IP 192.168.137.68
T = 32.19°C
T = 32.19°C
T = 31.73°C
T = 31.73°C
T = 31.73°C
T = 31.73°C
T = 30.79°C
```

<https://www.halvorsen.blog>



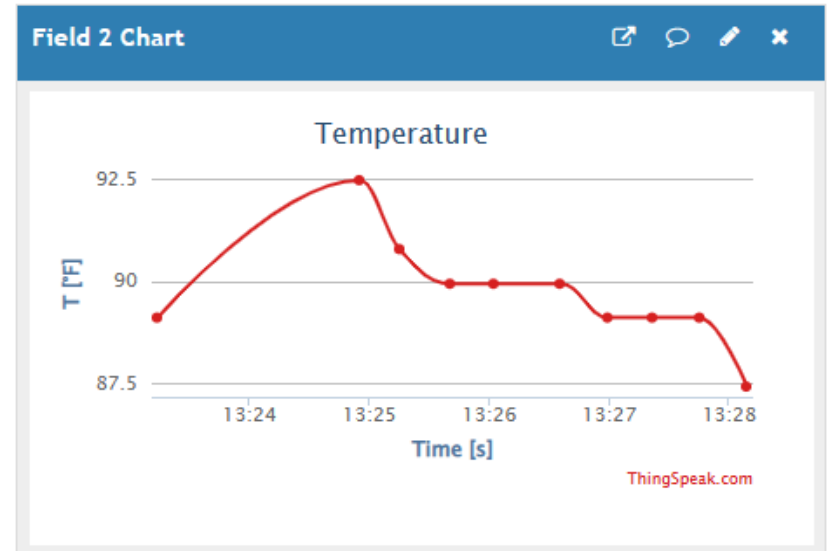
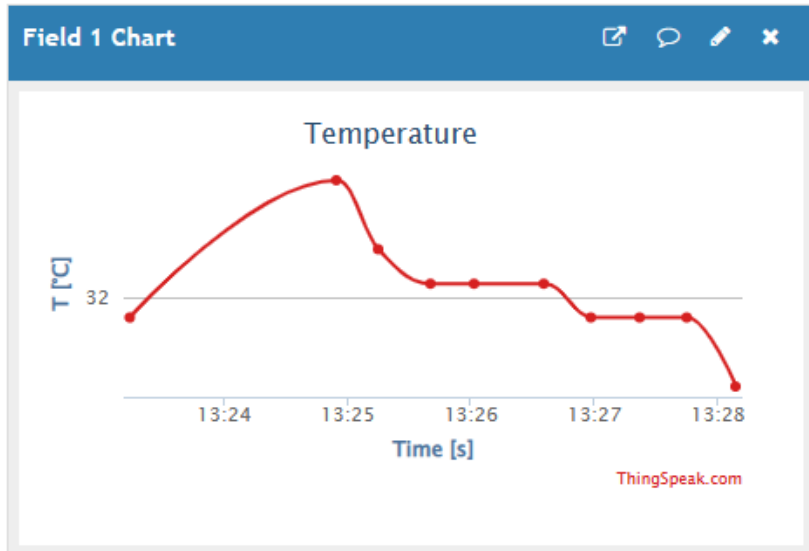
Final Application

Raspberry Pi Pico W and ThingSpeak IoT Cloud Service

Hans-Petter Halvorsen

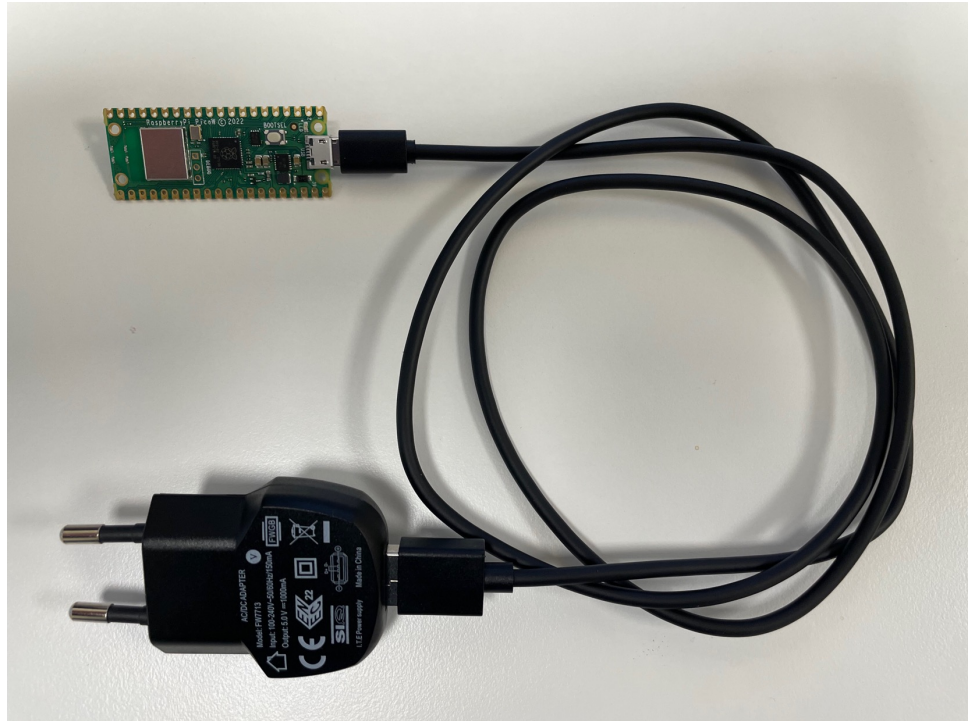
[Table of Contents](#)

ThingSpeak Website



Final System

Change name to **main.py** so that the program can run without a PC connected to the Raspberry Pi Pico



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](https://projects.raspberrypi.org/en/projects?hardware[]=pico)

- Getting Started with Pico:

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

- Getting started with your Raspberry Pi Pico W:

<https://projects.raspberrypi.org/en/projects/get-started-pico-w>

- MicroPython: <https://docs.micropython.org/en/latest/index.html>

Additional Resources

- ThingSpeak: <https://thingspeak.com>
- Send Sensor Data to Thingspeak with Raspberry Pi Pico W: <https://how2electronics.com/send-sensor-data-to-thingspeak-with-raspberry-pi-pico-w/>
- Python Requests post() Method: https://www.w3schools.com/python/ref_requests_post.asp
- MicroPython – network: <https://docs.micropython.org/en/latest/library/network.html#common-network-adapter-interface>
- MicroPython – urequests: https://makeblock-micropython-api.readthedocs.io/en/latest/public_library/Third-party-libraries/urequests.html

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: <https://www.halvorsen.blog>

