

<https://www.halvorsen.blog>



# OPC UA in Python

Hans-Petter Halvorsen



# Contents

- Introduction
- OPC UA Server Simulator from Integration Objects
- UaExpert OPC UA Client
- Python OPC UA Client Examples
- Python OPC UA Server Examples
- OPC UA Python Client GUI

<https://www.halvorsen.blog>

# Introduction

Hans-Petter Halvorsen



[Table of Contents](#)

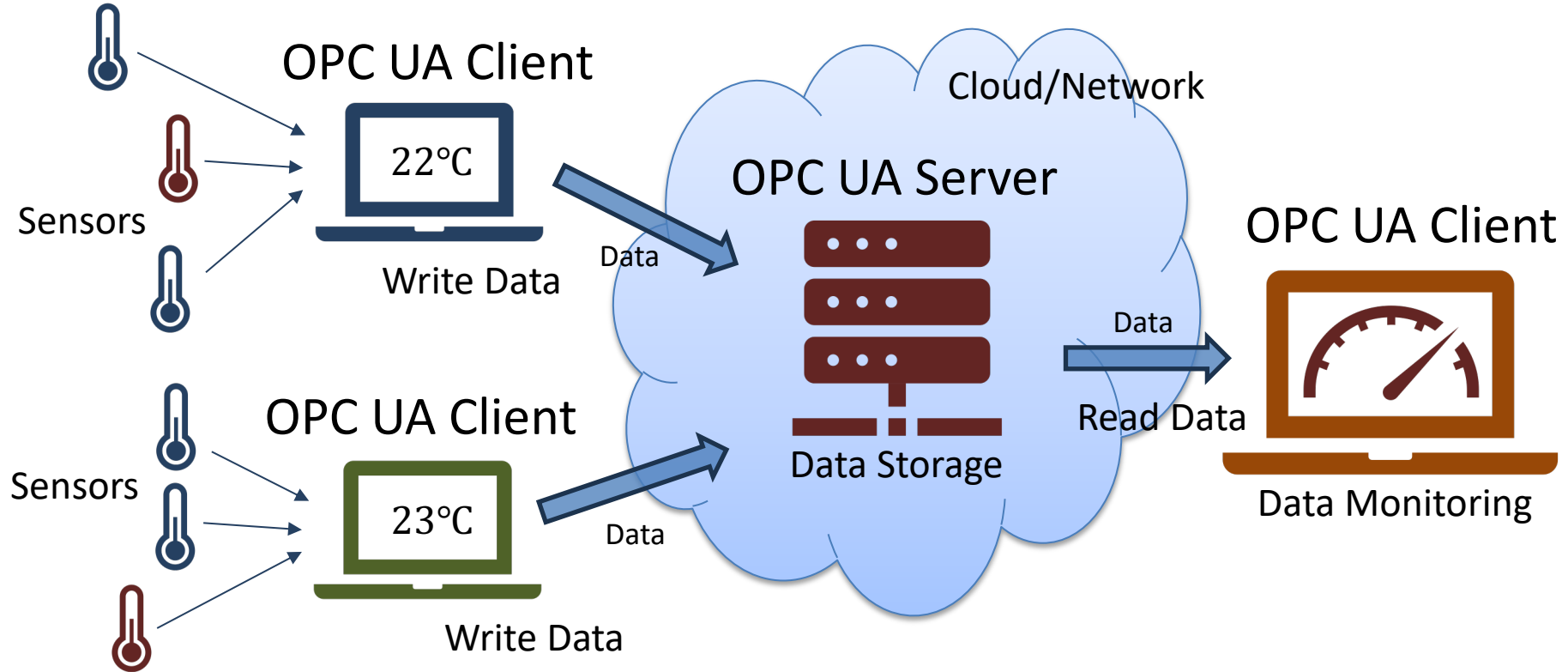
# Introduction

- This Tutorial will use the OPC UA protocol to send data between different devices or different programs.
- OPC UA is a communication protocol.
- OPC UA is mainly used for industrial purposes (industrial automation), and it is one of the most used protocols in the industry today.
- OPC UA is maintained by: <https://opcfoundation.org>.
- Python is a widely used multi-purpose Programming language and one of the most used programming languages today.
- Python is maintained by: <https://www.python.org>.



# OPC UA Example

We will create different OPC UA Clients and an OPC UA Server in this Tutorial



# Introduction

- There exists lots of different OPC packages and libraries for Python.
- This Tutorial will demonstrate OPC UA in Python with some basic Code Examples using the “**opcua**” Python package.
- This Tutorial will use the **Thonny** Python Editor and the **Spyder** Python Editor, but other Python Editors can of course also be used.
- This Tutorial will use the **OPC UA Server Simulator** from Integration Objects (free OPC UA server for Test and Demonstration purposes), but other OPC UA Servers can of course also be used.
- We will create both OPC UA Clients and an OPC UA Server using Python and the “**opcua**” Python package.

# Software

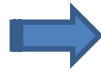
The following software is used in this Tutorial:

- Thonny Python Editor  
<https://thonny.org>
- Spyder Python Editor. You can download it separately or part of Anaconda Python Distribution  
<https://www.spyder-ide.org>  
<https://www.anaconda.com>
- Python package/library opcua  
<https://pypi.org/project/opcua/>
- OPC UA Server Simulator:  
<https://integrationobjects.com/sioth-opc/sioth-opc-unified-architecture/opc-ua-server-simulator/>
- UaExpert OPC UA Client:  
<https://www.unified-automation.com/products/development-tools/uaexpert.html>

# OPC Specifications

OPC is a standard that defines the communication of data between devices from different manufactures. OPC requires an **OPC Server** that communicates with one or more **OPC Clients**

“Classic” OPC

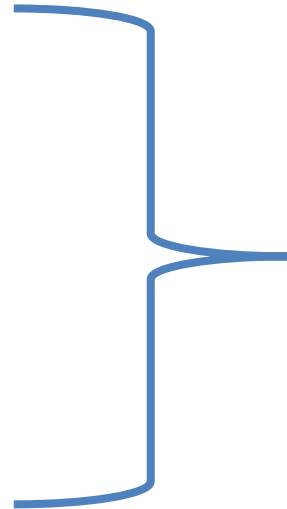


“Next Generation” OPC

OPC DA

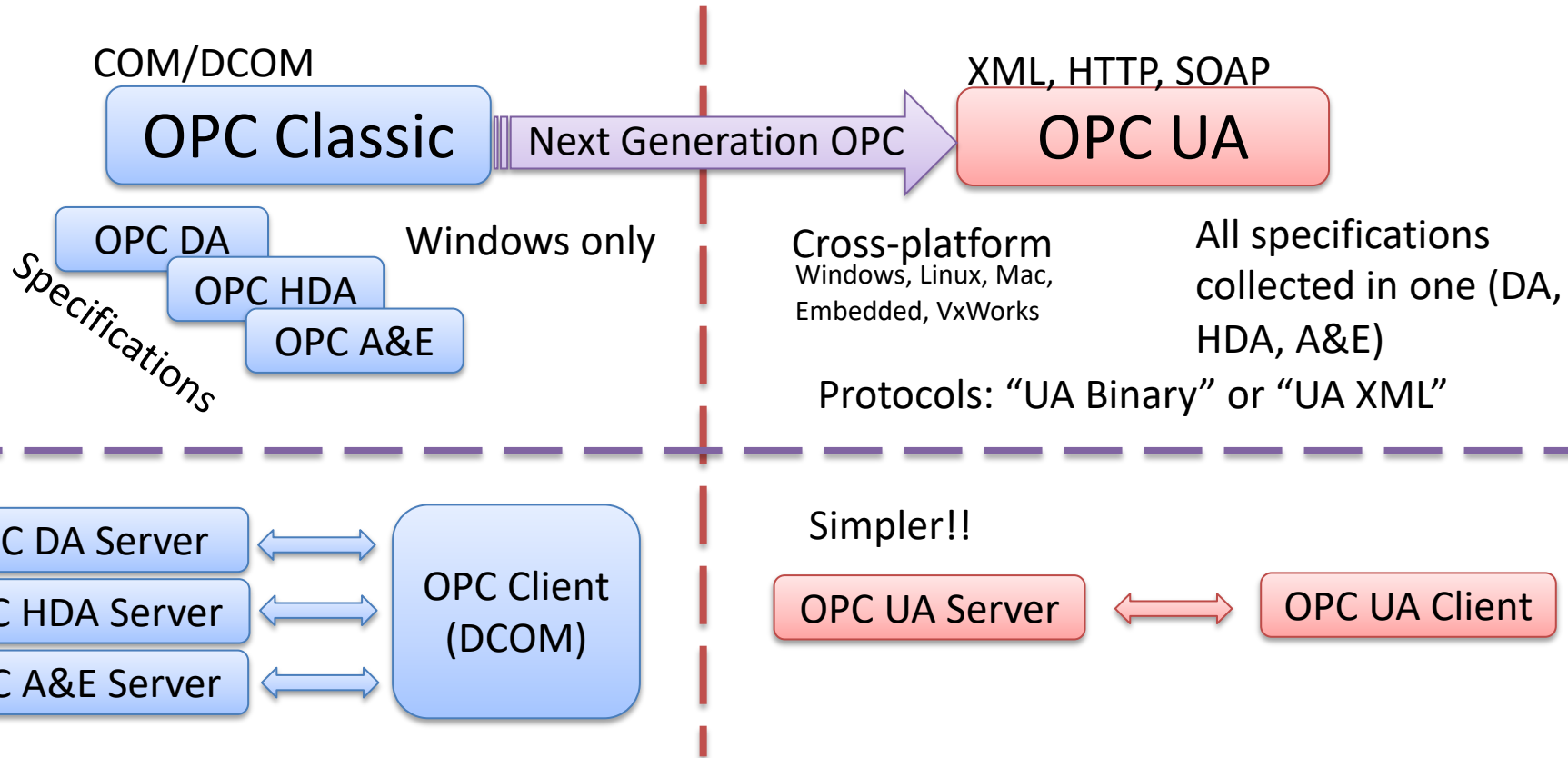
OPC HDA

OPC A&E



OPC UA

# Next Generation OPC



# OPC UA – Server and Clients

OPC is a standard that defines the communication of data between devices from different manufactures. OPC requires an **OPC Server** that communicates with one or more **OPC Clients**.

OPC UA Client



Data



OPC UA Server

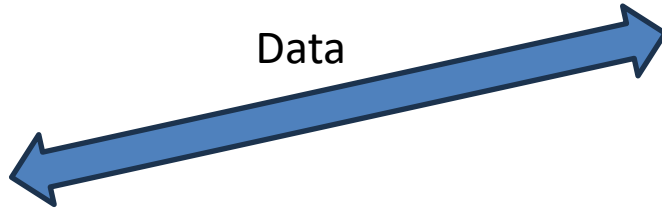


Data Storage

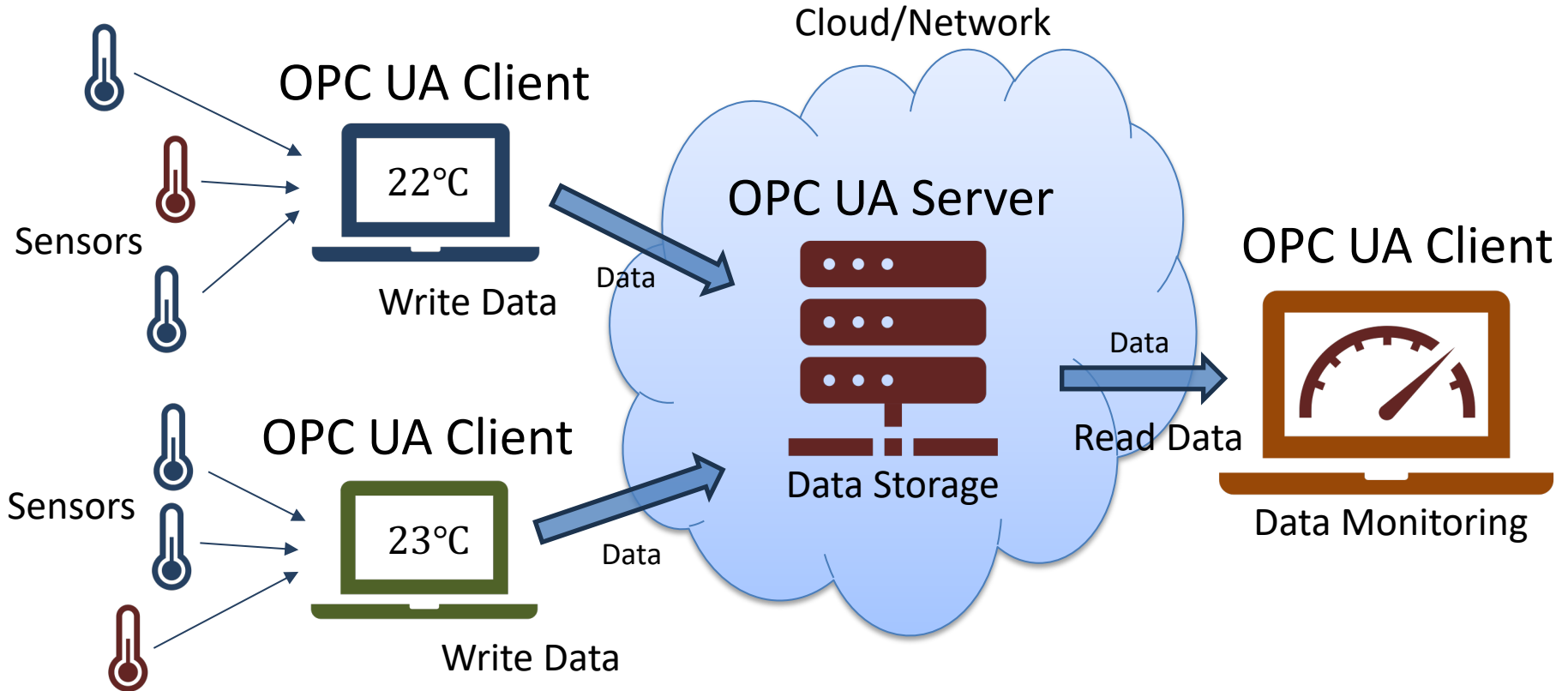
OPC UA Client



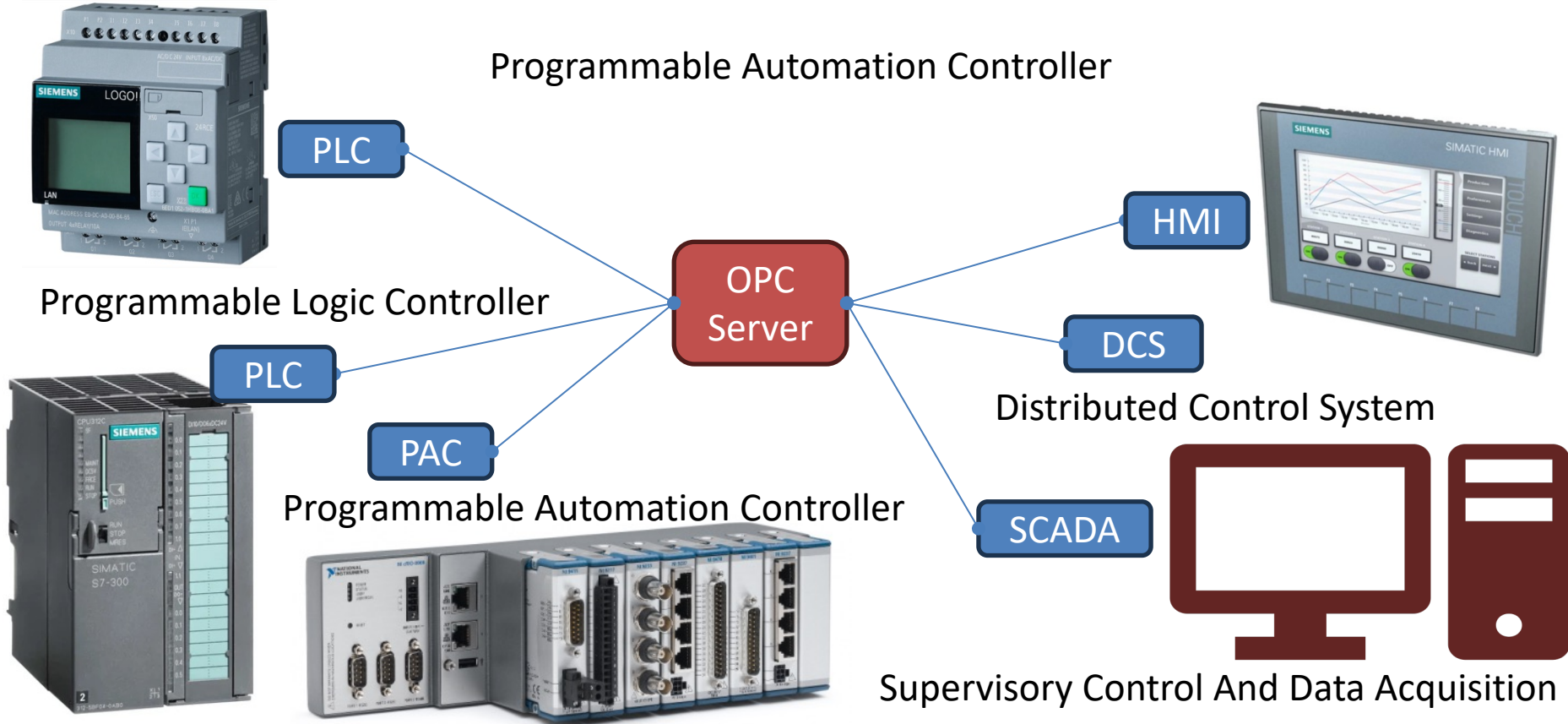
Data



# OPC UA Scenario

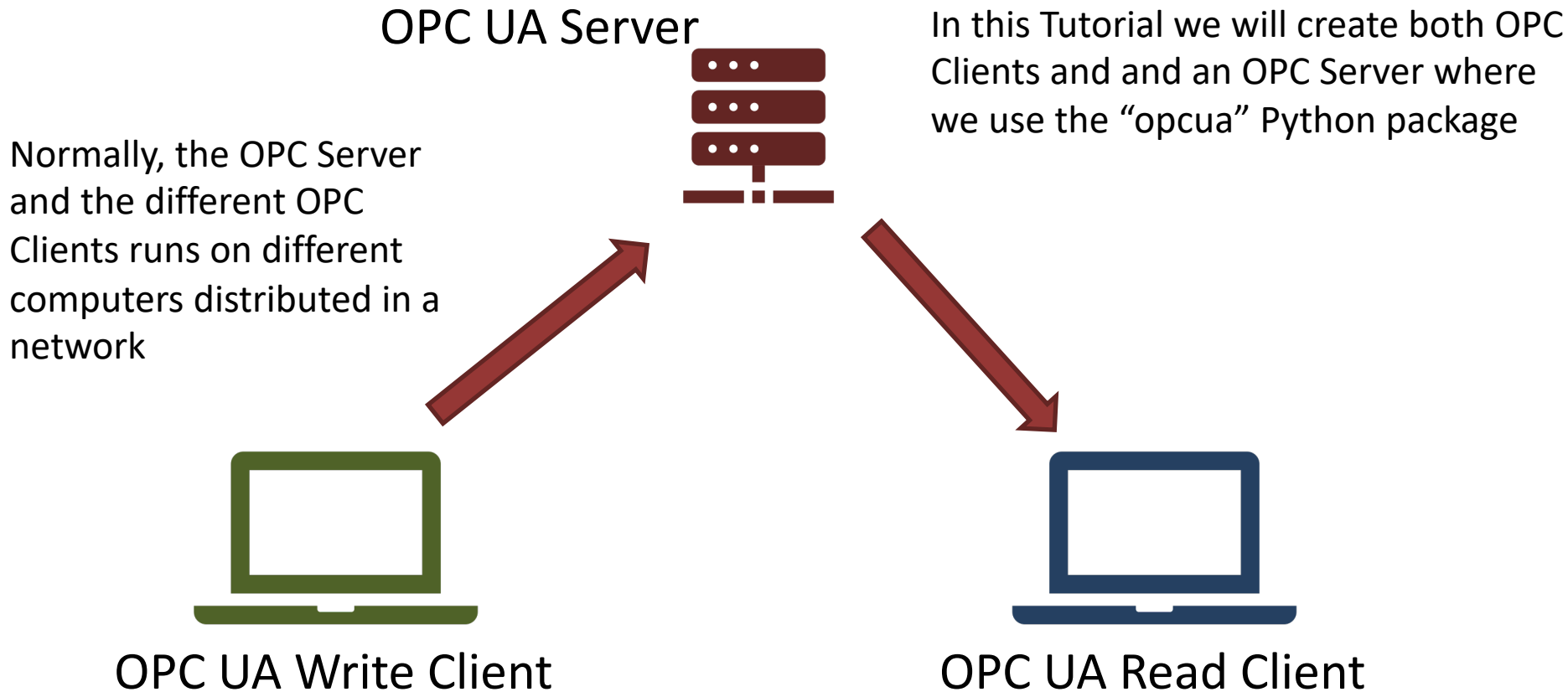


# OPC UA Scenario





# OPC UA Examples in this Tutorial



# Installation of “opcua” Thonny

The image displays two screenshots of the Thonny package manager interface, showing the search and details for the 'opcua' package.

**Left Screenshot: Search Results**

- Search bar: `opcua`
- Search button: `Search on PyPI`
- Search results list (left sidebar):
  - `<INSTALL>`
  - `adafruit-board-toolkit`
  - `astroid`
  - `asttokens`
  - `bcrypt`
  - `bitstring`
  - `cff`
  - `colorama`
  - `contourpy`
  - `cryptography`
  - `cycler`
  - `deprecation`
  - `dill`
  - `dnspython`
  - `docutils`
  - `ecdsa`
  - `esptool`
  - `fonttools`
  - `guizero`
  - `isort`
  - `jedi`
  - `kiwisolver`
  - `lazy-object-proxy`
- Search results list (main area):
  - `opcua`**  
Pure Python OPC-UA client and server library
  - `opcua-webclient`  
A OPCUA web client, implemented by Python.
  - `opcua-widgets`  
OPC-UA Widgets
  - `opcua-modeler`  
OPC-UA Address Space Modeler
  - `formic-opcua`  
<No description>
  - `opcua-client`  
OPC-UA Client GUI
  - `opcua-tools`

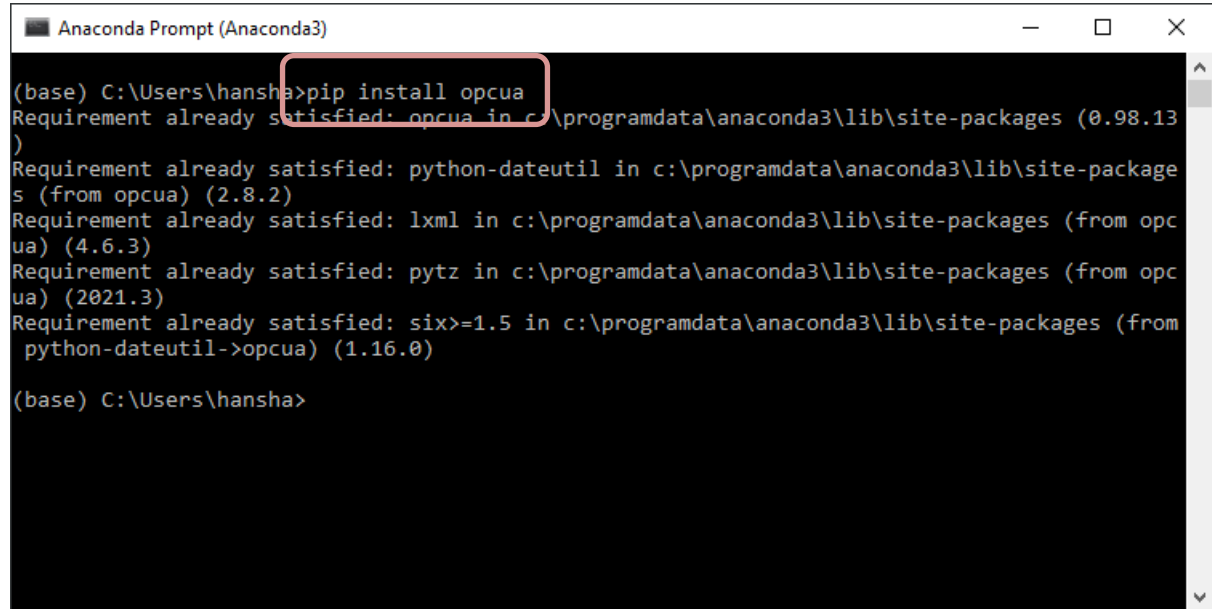
**Right Screenshot: Package Details**

- Search bar: `opcua`
- Search button: `Search on PyPI`
- Package details (main area):
  - `opcua`**
  - Installed version:** 0.98.13
  - Installed to:** `C:\Users\hansha\AppData\Roaming\Python\Python310\site-packages`
  - Latest stable version:** 0.98.13
  - Summary:** Pure Python OPC-UA client and server library
  - Author:** Olivier Roulet-Dubonnet
  - Homepage:** <http://freeopcua.github.io/>
  - PyPI page:** <https://pypi.org/project/opcua/>
- Package list (left sidebar):
  - `isort`
  - `jedi`
  - `kiwisolver`
  - `lazy-object-proxy`
  - `lxml`
  - `matplotlib`
  - `mccabe`
  - `mypy`
  - `mypy-extensions`
  - `nidaqmx`
  - `numpy`
  - `opcua`**
  - `packaging`
  - `paho-mqtt`
  - `paramiko`
  - `parso`
  - `pillow`
  - `pip`
  - `platformdirs`
  - `pyparser`
  - `pylint`
  - `pymongo`
  - `pynacl`
- Buttons at the bottom: `Upgrade`, `Uninstall`, `...`, `Close`

# Installation of “opcua” Anaconda

You can use the “**Anaconda Prompt**” that comes with the Anaconda Distribution

`pip install opcua`



```
Anaconda Prompt (Anaconda3)

(base) C:\Users\hansha>pip install opcua
Requirement already satisfied: opcua in c:\programdata\anaconda3\lib\site-packages (0.98.13)
Requirement already satisfied: python-dateutil in c:\programdata\anaconda3\lib\site-packages (from opcua) (2.8.2)
Requirement already satisfied: lxml in c:\programdata\anaconda3\lib\site-packages (from opcua) (4.6.3)
Requirement already satisfied: pytz in c:\programdata\anaconda3\lib\site-packages (from opcua) (2021.3)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil->opcua) (1.16.0)

(base) C:\Users\hansha>
```

<https://www.anaconda.com/download>

<https://pypi.org/project/opcua/>

# References

- PyPi:  
<https://pypi.org/project/opcua/>
- Python OPC-UA Documentation:  
<https://python-opcua.readthedocs.io/en/latest/>
- GitHub:  
<https://github.com/FreeOpcUa/python-opcua/tree/master/examples>
- GitHub Client GUI:  
<https://github.com/FreeOpcUa/opcua-client-gui>

<https://www.halvorsen.blog>

# OPC UA Simulation Server

Integration Objects

Hans-Petter Halvorsen

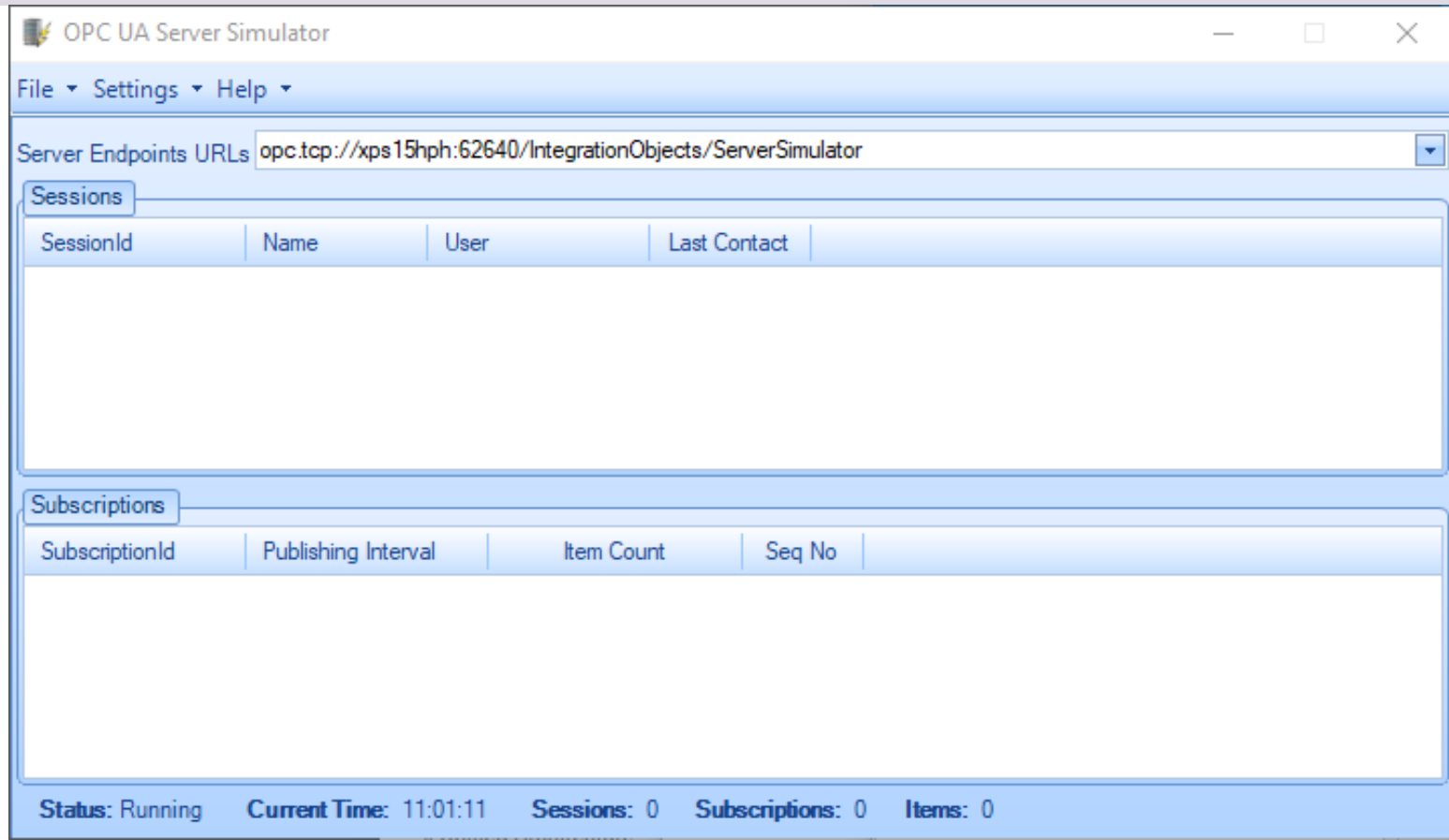


[Table of Contents](#)

# OPC UA Server Simulator

- “OPC UA Server Simulator” from Integration Objects is a free OPC UA Server
- You can use it for Test and Demonstration purposes.
- It runs 48 hours before you need to restart it
- Users can configure their own OPC Tags via CSV files
- Download:  
<https://integrationobjects.com/sioth-opc/sioth-opc-unified-architecture/opc-ua-server-simulator/>

# OPC UA Server Simulator

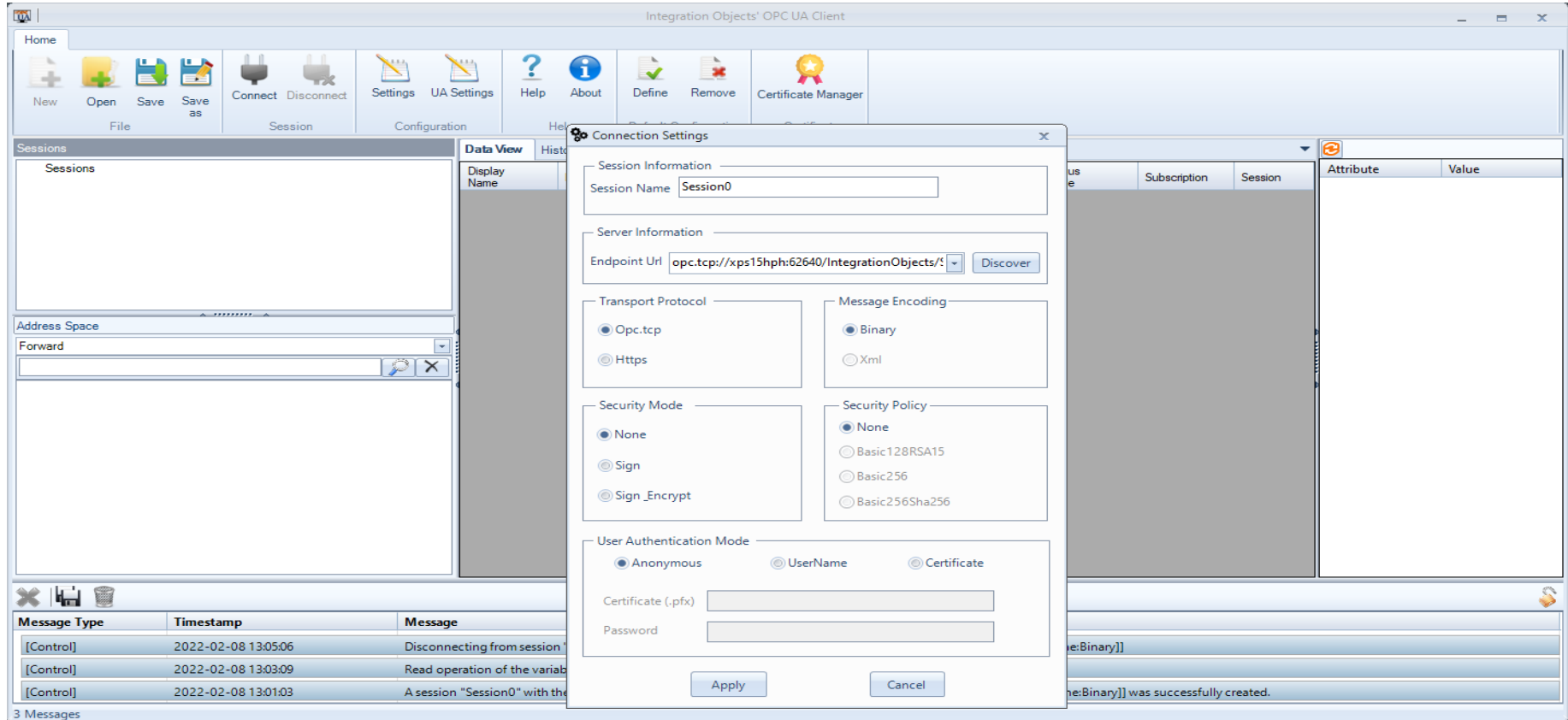


# OPC UA Client

- Integration Objects also offer a free OPC UA Client
- You can use it for Test and Demonstration purposes
- Download:  
<https://integrationobjects.com/sioth-opc/sioth-opc-unified-architecture/opc-ua-client/>



# OPC UA Client



# UaExpert OPC UA Client

Unified Automation

Hans-Petter Halvorsen



# UaExpert

- UaExpert is an OPC UA Client from Unified Automation
- Homepage:  
<https://www.unified-automation.com>
- Download UaExpert:  
<https://www.unified-automation.com/downloads/opc-ua-clients.html>

# UaExpert OPC UA Client

Unified Automation UaExpert - The OPC Unified Architecture Client - NewProject\*

File View Server Document Settings Help

Project

- Project
  - Servers
    - LabVIEW OPC UA Server
    - Python OPC UA Server
    - OPC UA Server Simulator from I...
  - Documents
    - Data Access View
    - Data Access View-1

Address Space

No Highlight

- Tag5
- Tag6
- Tag7
- Tag8
- Tag9
- Server
  - Types
  - Views

Data Access View

#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp
1	LabVIEW OPC U...	NS2[String]...	Temperature	26	Double	01:00:00.000	14:25:23.235
2	Python OPC UA...	NS2[Numeric]2	Temperature	20	Int64	01:00:00.000	01:00:00.000
3	OPC UA Server ...	NS2[String]Tag7	Tag7	22	Double	14:39:03.638	14:39:03.639

Attributes

Attribute

Value

- Node
  - NamespaceIndex: 2
  - IdentifierType: String
  - Identifier: Tag7
  - NodeClass: Variable
  - BrowseName: 2, "Tag7"
  - DisplayName: "en", "Tag7"
  - Description: Invalid Datatype: Ex
- Value
  - SourceTimestamp: 2023-12-05 14:39:03
  - SourcePicoSeconds: 0
  - ServerTimestamp: 2023-12-05 14:39:12
  - ServerPicoSeconds: 0
  - StatusCode: Good (0x00000000)

References

Forward

Reference: Target DisplayName

HasTypeDefiniti... BaseDataVariableType

Log

Timestamp	Source	Server	Message
2023-12-05 14:3...	DA Plugin		QasDaModel::dropMimeData
2023-12-05 14:3...	DA Plugin	OPC UA Server ...	No subscription available for ServerId 2
2023-12-05 14:3...	DA Plugin	OPC UA Server ...	Creating new subscription: ClientHandle=3, PublishingEnable=1, LifeTimeCount=2400, MaxKeepAliveCount=10, Priority=0, PublishingInterval=500, MaxNotificationsPerPublish=0
2023-12-05 14:3...	DA Plugin	OPC UA Server ...	CreateSubscription succeeded [ret = Good]
2023-12-05 14:3...	DA Plugin	OPC UA Server ...	Revised values: LifeTimeCount=2400, MaxKeepAliveCount=10, Priority=0, PublishingInterval=500, SubscriptionId=3
2023-12-05 14:3...	DA Plugin	OPC UA Server ...	Created subscription for ServerId 2
2023-12-05 14:3...	DA Plugin	OPC UA Server ...	Item [NS2[String]Tag7]: SamplingInterval=250, QueueSize=1, DiscardOldest=1, ClientHandle=7
2023-12-05 14:3...	DA Plugin	OPC UA Server ...	CreateMonitoredItems succeeded [ret = Good]
2023-12-05 14:3...	DA Plugin	OPC UA Server ...	Item [NS2[String]Tag7] succeeded : RevisedSamplingInterval=250, RevisedQueueSize=1, MonitoredItemId=3 [ret = Good]
2023-12-05 14:3...	Attribute Plugin	OPC UA Server ...	Read attributes of node 'NS2[String]Tag7' succeeded [ret = Good].

# Python OPC UA Client Examples

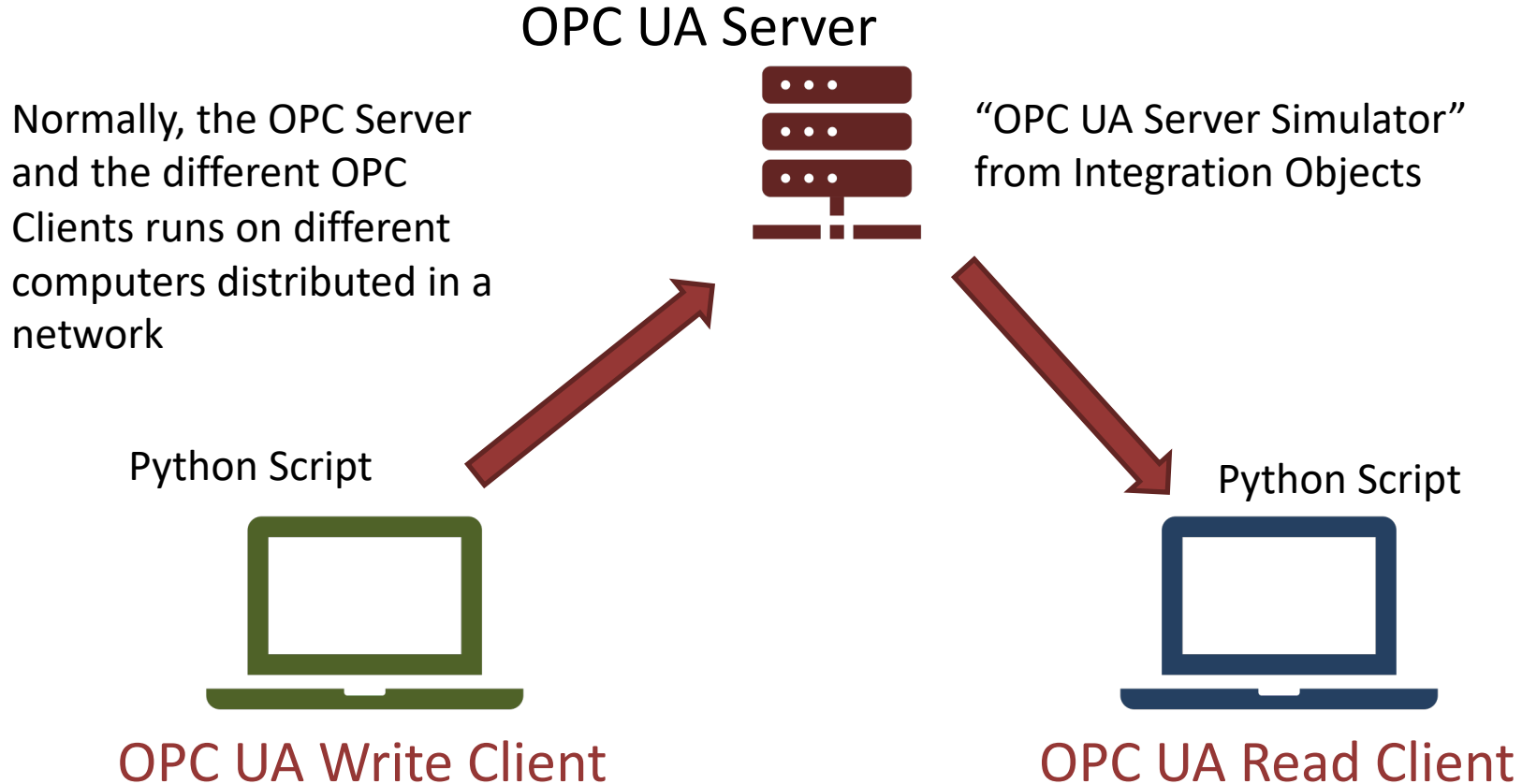
Hans-Petter Halvorsen



# Python OPC UA Client Examples

- We start with creating OPC UA Clients with Python
- Typically, most of the times we need to create OPC UA clients, while we use existing OPC UA Servers from different vendors that already exists in our system
- Here, we will use the “OPC UA Server Simulator” from Integration Objects for Testing our OPC UA Python examples

# Python OPC UA Client Examples



# OPC UA Client Write

```
from opcua import Client
```

```
url = "xxx"
```

```
nodeId = "xxx"
```

```
client = Client(url)
```

```
client.connect()
```

```
node = client.get_node(nodeId)
```

```
value = float(20)
```

```
node.set_data_value(value)
```

```
client.disconnect()
```



# OPC UA Client Read

```
from opcua import Client

url = "xxx"
nodeId = "xxx"

client = Client(url)
client.connect()

node = client.get_node(nodeId)
value = node.get_value()
print(value)

client.disconnect()
```

# Python OPC UA Client Examples

The image displays three windows from the Thonny Python IDE. At the top is the 'OPC UA Server Simulator' window, which shows the 'Server Endpoints URLs' as 'opc.tcp://xps15ph:62640/IntegrationObjects/ServerSimulator' and an empty 'Sessions' table. Below this are two Python script windows. The left window, 'opcua\_client\_write\_ex.py', contains code to connect to the server, get the node 'ns=2;s=Tag7', set its value to 22.0, and disconnect. A red box labeled 'Write Data to OPC UA Server' is overlaid on the bottom right of this script. The right window, 'opcua\_client\_read\_ex.py', contains code to connect to the server, get the node 'ns=2;s=Tag7', read its value, print it, and disconnect. A red box labeled 'Read Data from OPC UA Server' is overlaid on the bottom right of this script. Blue arrows point from the 'client.connect()' line in the write script to the 'Sessions' table in the simulator, and from the 'node.get\_value()' line in the read script to the same table. The shell windows at the bottom show the execution of the scripts, with the read script outputting '22.0'.

**OPC UA Server Simulator**

File Settings Help

Server Endpoints URLs: opc.tcp://xps15ph:62640/IntegrationObjects/ServerSimulator

Sessions

SessionId	Name	User	Last Contact
-----------	------	------	--------------

**opcua\_client\_write\_ex.py**

```
1 from opcua import Client
2
3 #OPC UA Server Simulator from Integration Objects
4 url = "opc.tcp://localhost:62640/IntegrationObjects/ServerSimulator/"
5 nodeId = "ns=2;s=Tag7"
6
7 client = Client(url)
8 client.connect()
9
10 node = client.get_node(nodeId)
11 value = float(22)
12 node.set_data_value(value)
13
14 client.disconnect()
```

**Write Data to OPC UA Server**

Shell

```
>>> %Run opcua_client_write_ex.py
>>>
```

**opcua\_client\_read\_ex.py**

```
1 from opcua import Client
2
3 #OPC UA Server Simulator from Integration Objects
4 url = "opc.tcp://localhost:62640/IntegrationObjects/ServerSimulator/"
5 nodeId = "ns=2;s=Tag7"
6
7 client = Client(url)
8 client.connect()
9
10 node = client.get_node(nodeId)
11 value = node.get_value()
12 print(value)
13
14 client.disconnect()
```

**Read Data from OPC UA Server**

Shell

```
>>> %Run opcua_client_read_ex.py
22.0
>>>
```

Local Python 3 • Thonny's Python

# OPC UA Client Write Example

```
from opcua import Client

#OPC UA Server Simulator from Integration Objects
url = "opc.tcp://localhost:62640/IntegrationObjects/ServerSimulator/"
nodeId = "ns=2;s=Tag7"

client = Client(url)
client.connect()

node = client.get_node(nodeId)
value = float(20)
node.set_data_value(value)

client.disconnect()
```

# OPC UA Client Read Example

```
from opcua import Client

#OPC UA Server Simulator from Integration Objects
url = "opc.tcp://localhost:62640/IntegrationObjects/ServerSimulator/"
nodeId = "ns=2;s=Tag7"

client = Client(url)
client.connect()

node = client.get_node(nodeId)
value = node.get_value()
print(value)

client.disconnect()
```

# Improvements

We improve the basic Write and Read Examples

- Adding While Loop, etc.
- Error Handling
- Getting Data from a real Sensor, e.g., a Temperature Sensor
  - I will create a “Simulator” that simulates a real Temperature Sensor using the Random Generator in Python

## OPC UA Client **Write** Example

```
from opcua import Client
import time
import random

#OPC UA Server Simulator from Integration Objects
url = "opc.tcp://localhost:62640/IntegrationObjects/ServerSimulator/"
nodeId = "ns=2;s=Tag7"

client = Client(url)
client.connect()
print("OPC UA Client Connected")
print("Press Ctrl-C to Stop Program")

try:
    while True:
        value = random.randint(20,30)
        print(value)

        node = client.get_node(nodeId)
        value = float(value)
        node.set_data_value(value)
        time.sleep(10)

except KeyboardInterrupt:
    pass

client.disconnect()
print("OPC UA Client Disconnected and Program Stopped")
```

## OPC UA Client **Read** Example

```
from opcua import Client
import time

#OPC UA Server Simulator from Integration Objects
url = "opc.tcp://localhost:62640/IntegrationObjects/ServerSimulator/"
nodeId = "ns=2;s=Tag7"

client = Client(url)
client.connect()
print("OPC UA Client Connected")
print("Press Ctrl-C to Stop Program")

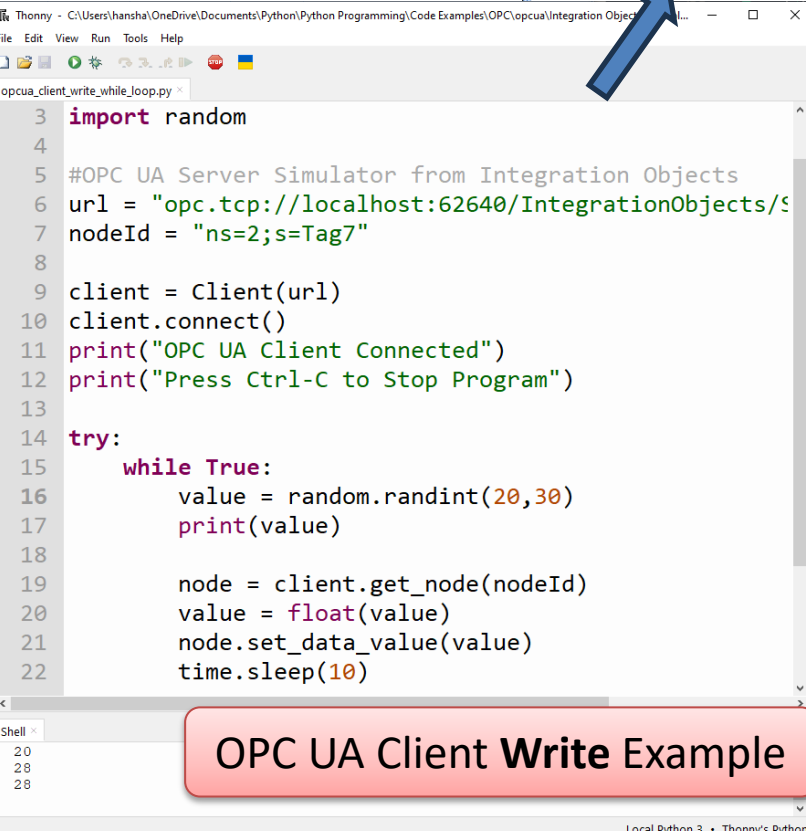
try:
    while True:
        node = client.get_node(nodeId)
        value = node.get_value()
        print(value)
        time.sleep(10)

except KeyboardInterrupt:
    pass

client.disconnect()
print("OPC UA Client Disconnected and Program Stopped")
```

# "OPC UA Server Simulator"

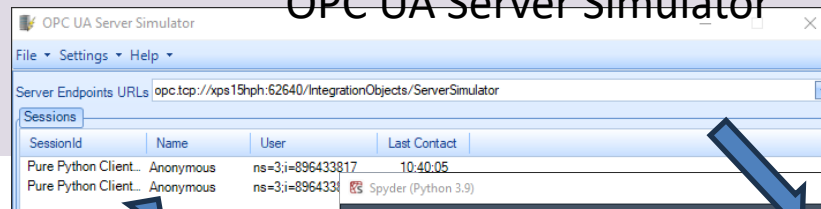
## Thonny Python Editor



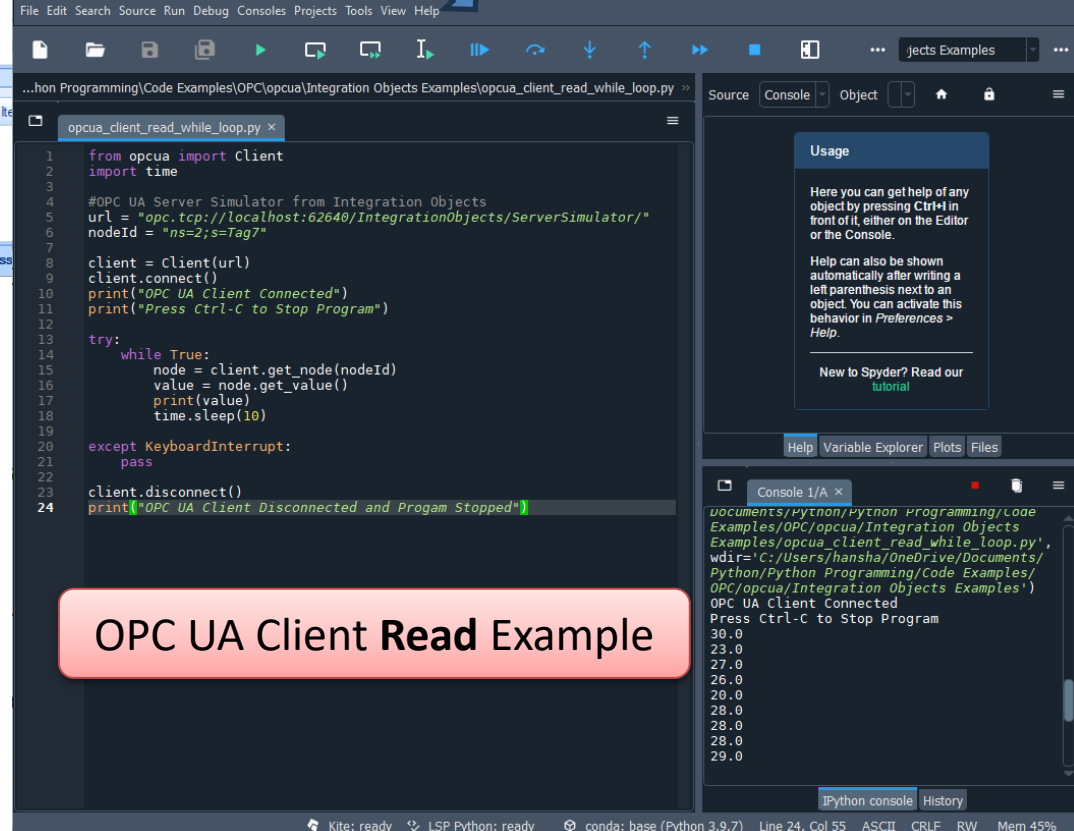
The screenshot shows the Thonny Python Editor with a file named `opcua_client_write_while_loop.py` open. The code is as follows:

```
3 import random
4
5 #OPC UA Server Simulator from Integration Objects
6 url = "opc.tcp://localhost:62640/IntegrationObjects/ServerSimulator/"
7 nodeId = "ns=2;s=Tag7"
8
9 client = Client(url)
10 client.connect()
11 print("OPC UA Client Connected")
12 print("Press Ctrl-C to Stop Program")
13
14 try:
15     while True:
16         value = random.randint(20,30)
17         print(value)
18
19         node = client.get_node(nodeId)
20         value = float(value)
21         node.set_data_value(value)
22         time.sleep(10)
```

At the bottom of the editor, there is a red callout box with the text "OPC UA Client Write Example".



## Spyder Python Editor



The screenshot shows the Spyder Python Editor with a file named `opcua_client_read_while_loop.py` open. The code is as follows:

```
1 from opcua import client
2 import time
3
4 #OPC UA Server Simulator from Integration Objects
5 url = "opc.tcp://localhost:62640/IntegrationObjects/ServerSimulator/"
6 nodeId = "ns=2;s=Tag7"
7
8 client = client(url)
9 client.connect()
10 print("OPC UA Client Connected")
11 print("Press Ctrl-C to Stop Program")
12
13 try:
14     while True:
15         node = client.get_node(nodeId)
16         value = node.get_value()
17         print(value)
18         time.sleep(10)
19
20 except KeyboardInterrupt:
21     pass
22
23 client.disconnect()
24 print("OPC UA Client Disconnected and Program Stopped")
```

At the bottom of the editor, there is a red callout box with the text "OPC UA Client Read Example".

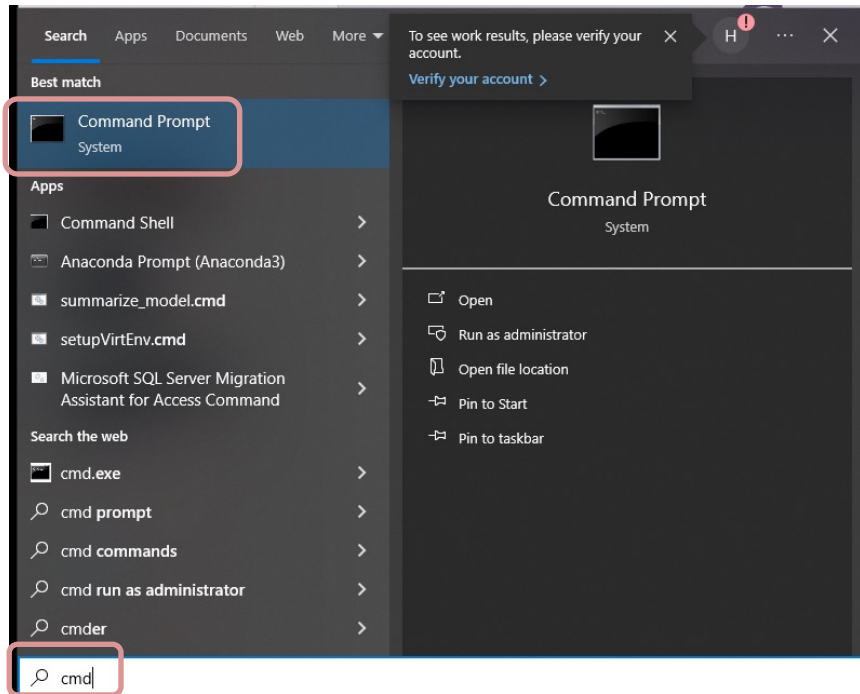
On the right side of the Spyder interface, there is a "Console" pane showing the output of the program:

```
documents\python\python_programming\code
Examples\opcua\Integration Objects
Examples\opcua_client_read_while_loop.py',
wdir='C:/Users/hansha/OneDrive/Documents/
Python/Python_Programming/Code_Examples/
OPC/opcua/Integration Objects Examples')
OPC UA Client Connected
Press Ctrl-C to Stop Program
30.0
23.0
27.0
26.0
20.0
28.0
28.0
29.0
```

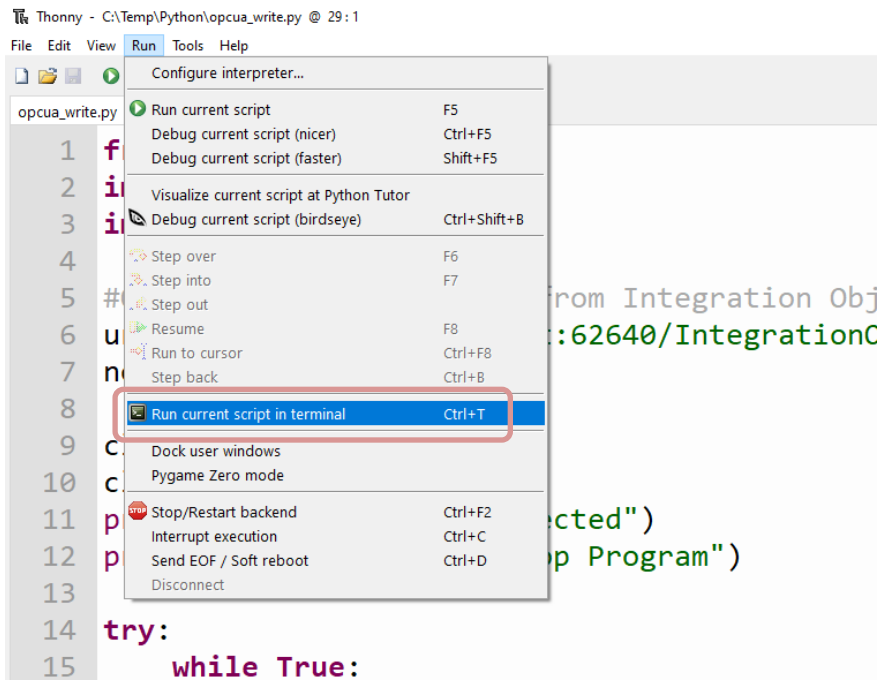


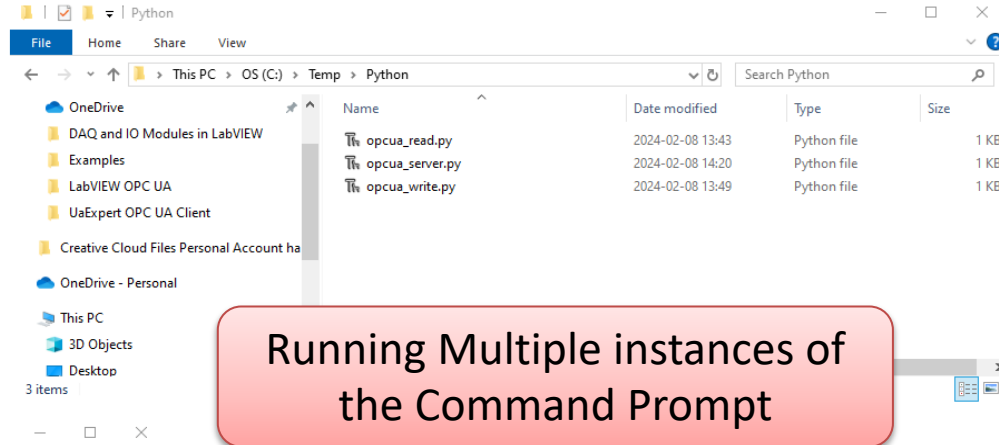
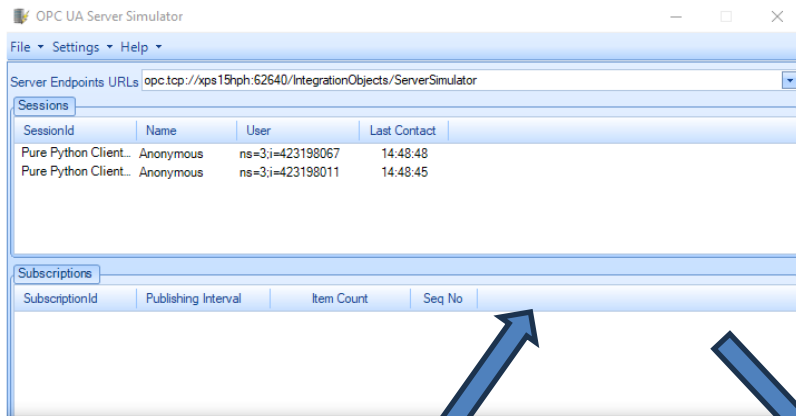
# Running Multiple instances of the Command Prompt

Open the Command Prompt from the Start menu



You can also do it from the Thonny Editor:





Running Multiple instances of the Command Prompt

```
cmd Command Prompt - python opcu_write.py
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\hansha>cd /
```

```
C:\>cd C:\Temp\Python
```

```
C:\Temp\Python>python opcu_write.py
cryptography is not installed, use of crypto disabled
cryptography is not installed, use of crypto disabled
OPC UA Client Connected
Press Ctrl-C to Stop Program
```

```
29
23
20
29
30
27
```

OPC UA Client **Write** Example

```
cmd Command Prompt - python opcu_read.py
```

```
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\hansha>cd /
```

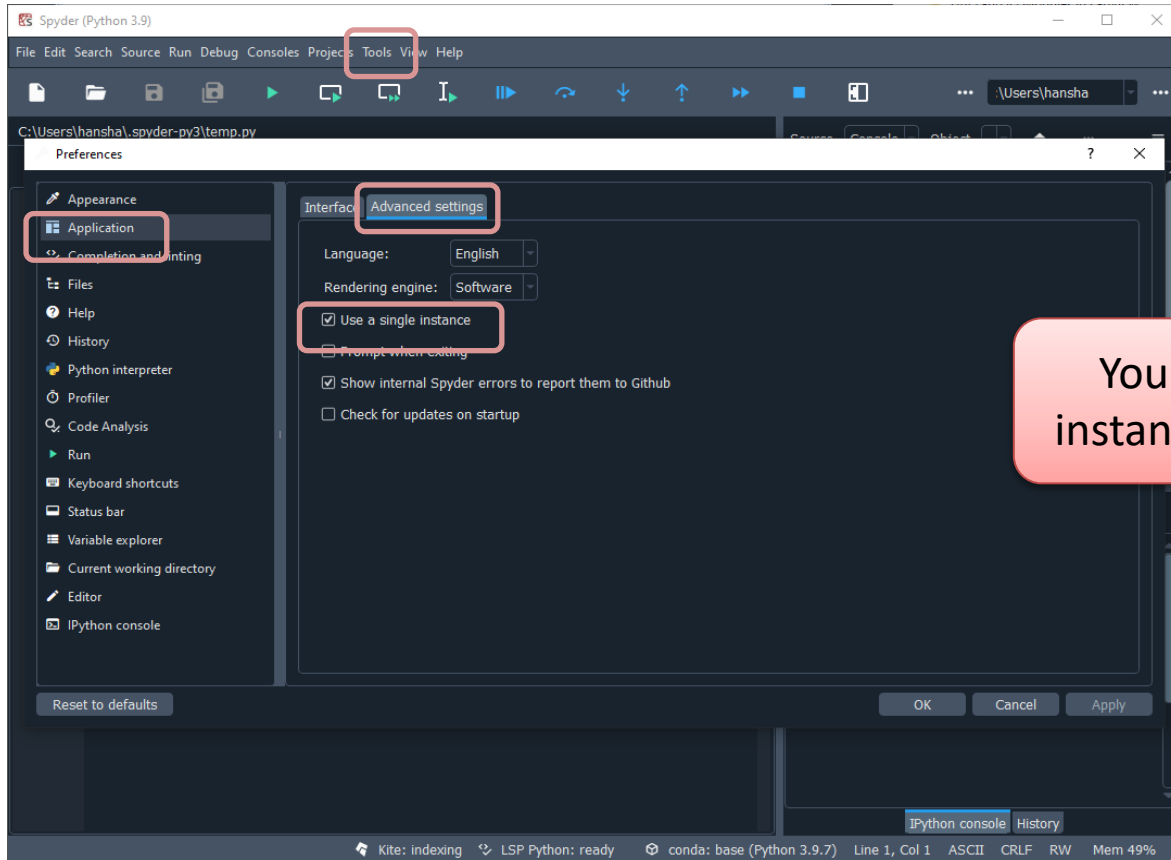
```
C:\>cd C:\Temp\Python
```

```
C:\Temp\Python>python opcu_read.py
cryptography is not installed, use of crypto disabled
cryptography is not installed, use of crypto disabled
OPC UA Client Connected
Press Ctrl-C to Stop Program
```

```
29.0
23.0
20.0
29.0
30.0
27.0
```

OPC UA Client **Read** Example

# Multiple Spyder Editors



You can also run Multiple instances of the Spyder Editor

## OPC UA Client Read Example

OPC UA Server Simulator

File Settings Help

Server Endpoints URLs `opc.tcp://xps15hp.62640/IntegrationObjects/ServerSimulator`

Sessions

SessionId	Name	User	Last Contact
Pure Python Client..	Anonymous	ns=3;i=221655076	15:08:51
Pure Python Client..	Anonymous	ns=3;i=221655132	15:08:57

Spyder (Python 3.9)

File Edit Search Source Run Debug Consoles Projects

C:\Temp\Python\opcua\_read.py

```
opcua_read.py x
1 from opcua import Client
2 import time
3
```

```
Simulator from Integration Objects
//localhost:62640/IntegrationObjects/ServerSimulator/
s=Tag7"
```

```
(url)
Client Connected")
url-C to Stop Program")
```

```
client.get_node(nodeId)
= node.get_value()
(value)
Loop(10)
```

```
Interrupt:
ect()
Client Disconnected and Program Stopped")
```

Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this

Source Console Object

Help Variable Explorer Plots Files

Console 1/A x

```
In [1]: runfile('C:/Temp/Python/
opcua_write.py', wdir='C:/Temp/
Python')
OPC UA Client Connected
Press Ctrl-C to Stop Program
24
24
29
25
28
20
29
```

IPython console History

Lines 2, Col 12 AS400 OMPF PSW Release 5.1 P56

OPC UA Server Simulator

File Settings Help

Server Endpoints URLs `opc.tcp://xps15hp.62640/IntegrationObjects/ServerSimulator`

Sessions

SessionId	Name	User	Last Contact
Pure Python Client..	Anonymous	ns=3;i=221655076	15:08:51
Pure Python Client..	Anonymous	ns=3;i=221655132	15:08:57

Spyder (Python 3.9)

File Edit Search Source Run Debug Consoles Projects Tools View Help

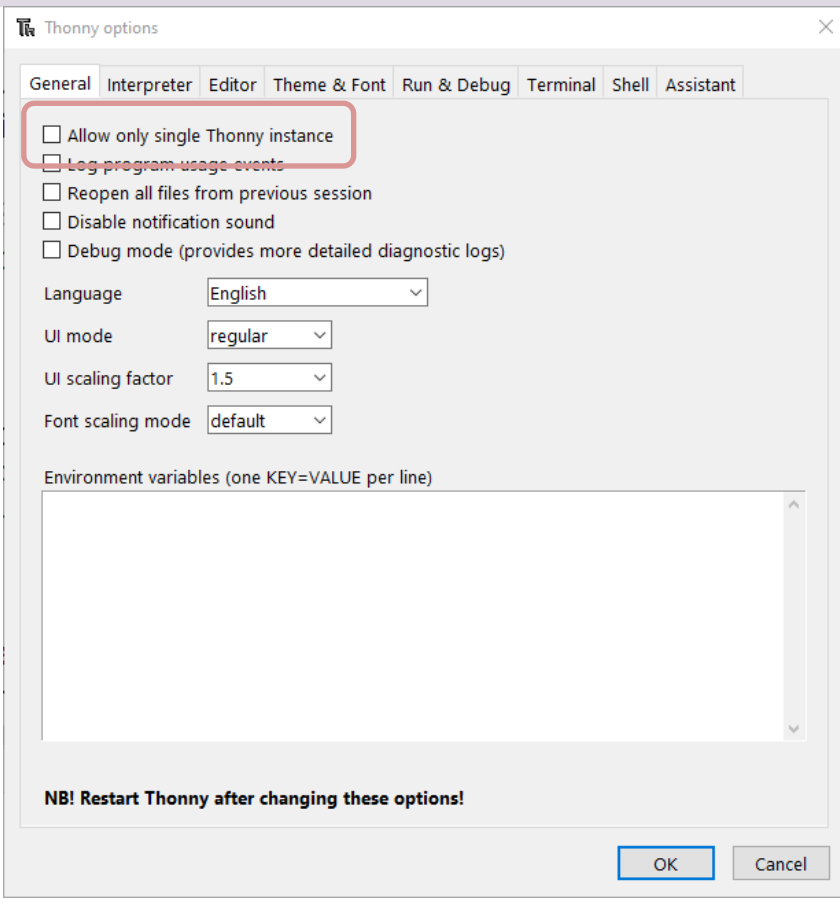
C:\Temp\Python\opcua\_write.py

```
opcua_write.py x
1 from opcua import Client
2 import time
3 import random
4
5 #OPC UA Server Simulator from Integration Objects
6 url = "opc.tcp://localhost:62640/IntegrationObjects/ServerSimulator/"
7 nodeId = "ns=2;s=Tag7"
8
9 client = Client(url)
10 client.connect()
11 print("OPC UA Client Connected")
12 print("Press Ctrl-C to Stop Program")
13
14 try:
15     while True:
16         value = random.randint(20,30)
17         print(value)
18
19         node = client.get_node(nodeId)
20         value = float(value)
21         node.set_data_value(value)
22         time.sleep(10)
23
24 except KeyboardInterrupt:
25     pass
26
27 client.disconnect()
28 print("OPC UA Client Disconnected and Program Stopped")
29
```

## OPC UA Client Write Example

## Running Multiple instances of the Spyder Editor

# Multiple Thonny Editors



Tools -> Options ...

You can also run Multiple instances of the Thonny Editor

## OPC UA Client Read Example

The image shows a screenshot of the OPC UA Server Simulator and the Thonny IDE. The OPC UA Server Simulator window is in the background, displaying the 'Sessions' tab with a table of active sessions. A blue arrow points from the 'Sessions' tab to the 'opcu\_read.py' script in the Thonny IDE. Another blue arrow points from the 'opcu\_write.py' script to the 'Sessions' tab.

**OPC UA Server Simulator Sessions Table:**

SessionId	Name	User	Last Contact
Pure Python Client...	Anonymous	ns=3;i=1297734378	10:47:25
Pure Python Client...	Anonymous	ns=3;i=1297734434	10:47:28

**Thonny IDE - opcu\_read.py:**

```
1 from opcua import Client
2 import time
3
4 #OPC UA Server Simulator from Integration Objects
5 url = "opc.tcp://localhost:62640/IntegrationObjects/ServerSim
6 nodeId = "ns=2;s=Tag7"
7
8 client = Client(url)
9 client.connect()
10 print("OPC UA Client Connected")
11 print("Press Ctrl-C to Stop Program")
12
13 try:
14     while True:
15         node = client.get_node(nodeId)
```

**Thonny IDE - opcu\_write.py:**

```
1 from opcua import Client
2 import time
3 import random
4
5 #OPC UA Server Simulator from Integration Objects
6 url = "opc.tcp://localhost:62640/IntegrationObjects/ServerSim
7 nodeId = "ns=2;s=Tag7"
8
9 client = Client(url)
10 client.connect()
11 print("OPC UA Client Connected")
12 print("Press Ctrl-C to Stop Program")
13
14 try:
15     while True:
16         node = client.get_node(nodeId)
```

**Thonny IDE Shell:**

```
>>> %Run opcu_write.py
OPC UA Client Connected
Press Ctrl-C to Stop Program
28.0
```

**Running Multiple instances of the Thonny Editor**

**OPC UA Client Write Example**

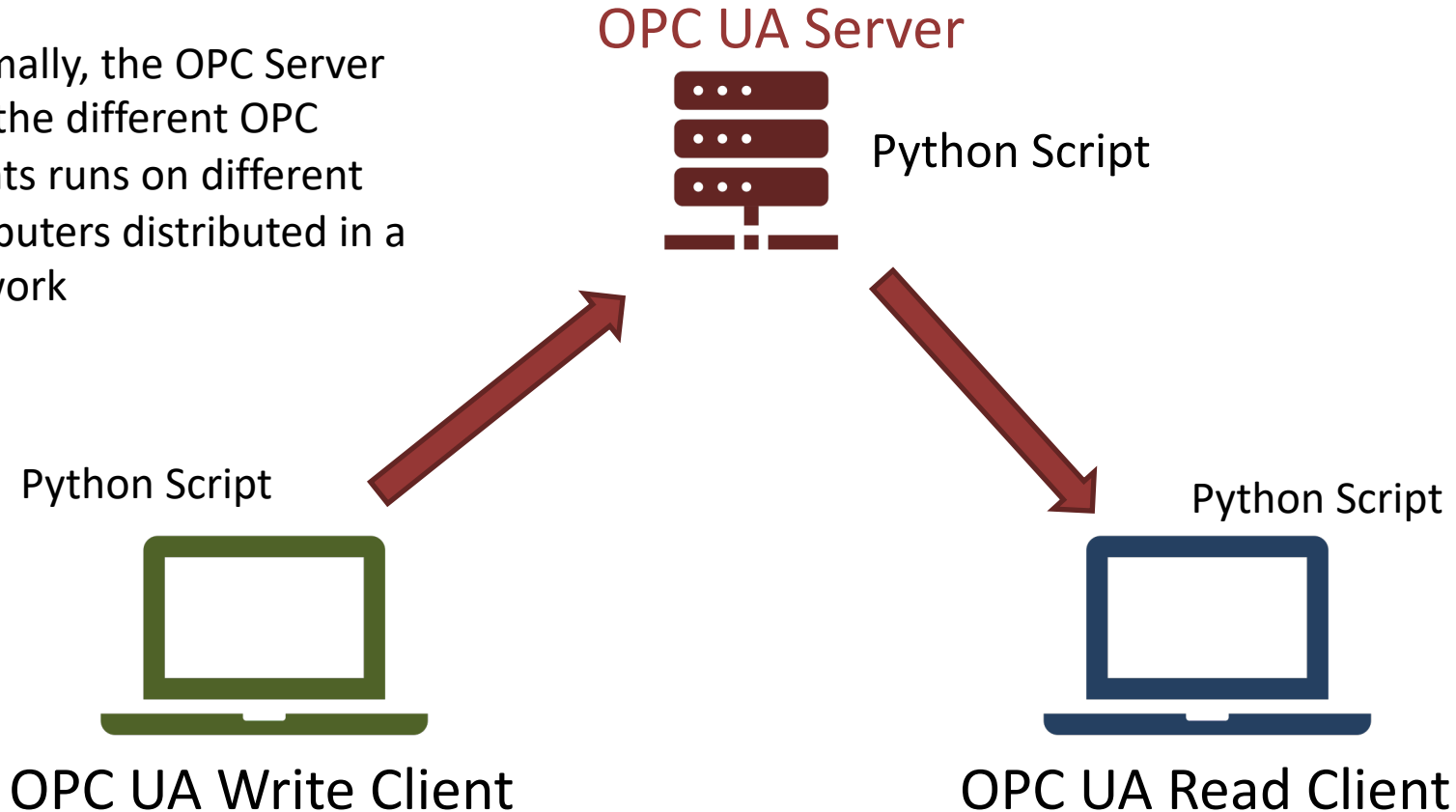
# Python OPC UA Server Examples

Hans-Petter Halvorsen



# Python OPC UA Server

Normally, the OPC Server and the different OPC Clients runs on different computers distributed in a network





# OPC UA Server

```
from opcua import Server
```

This is the main Python code  
for creating an OPC UA Server

```
server = Server()
```

```
server_url = "opc.tcp://127.0.0.1:1234"
```

```
server.set_endpoint(server_url)
```

```
name = "opcuapython"
```

```
namespace = server.register_namespace(name)
```

```
node = server.get_objects_node()
```

```
param = node.add_object(namespace, "Sensors")
```

```
var = param.add_variable(namespace, "Temperature", 0)
```

```
var.set_writable()
```

```
server.start()
```

# OPC UA Server

```
from opcua import Server
import time
from datetime import datetime

server = Server()
server_url = "opc.tcp://127.0.0.1:1234"
server.set_endpoint(server_url)

name = "opcuapython"
namespace = server.register_namespace(name)
node = server.get_objects_node()
param = node.add_object(namespace, "Sensors")
var = param.add_variable(namespace, "Temperature", 0)
var.set_writable()

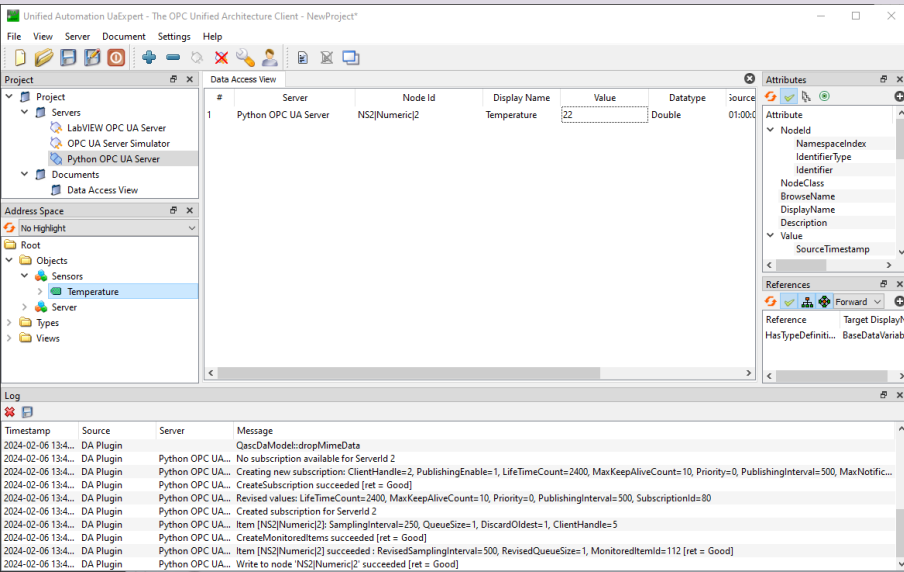
server.start()
print("OPC UA Server Started")
print("Press Ctrl-C to Stop Program")

try:
    while True:
        now = datetime.now()
        current_time = now.strftime("%H:%M:%S")
        print("OPC UA Server Running", current_time)
        value = var.get_value()
        print("Current Value:", value)
        time.sleep(1)

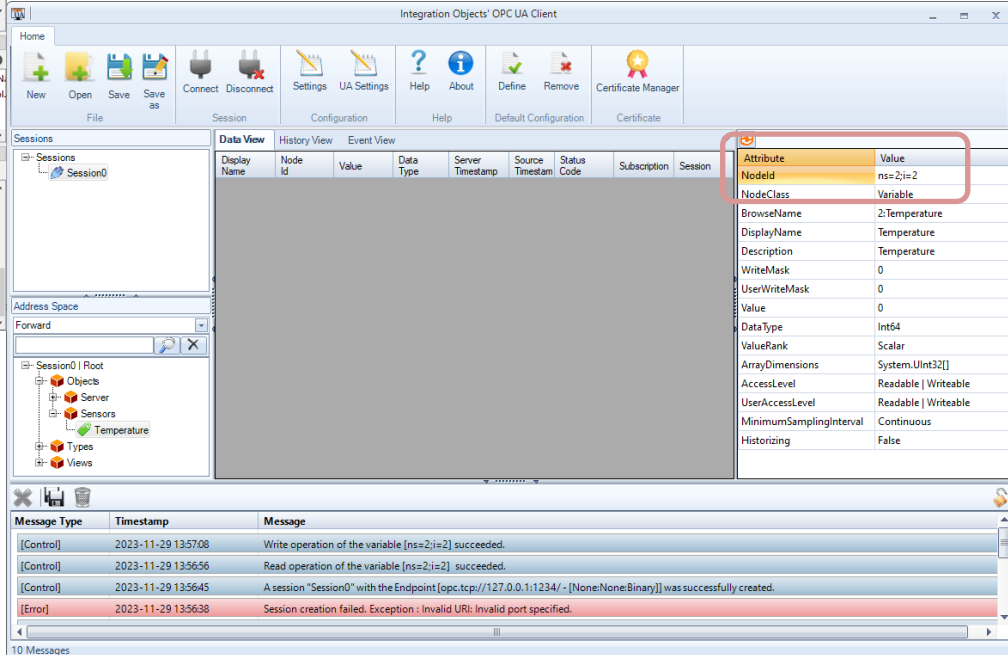
except KeyboardInterrupt:
    pass

server.stop()
print("OPC UA Server Stopped")
```

# Test Connection to Server



## OPC UA Client from Integration Objects



UaExpert OPC UA Client

# Get NodeId

```
from opcua import Client

#OPC UA Server Python
url = "opc.tcp://127.0.0.1:1234"

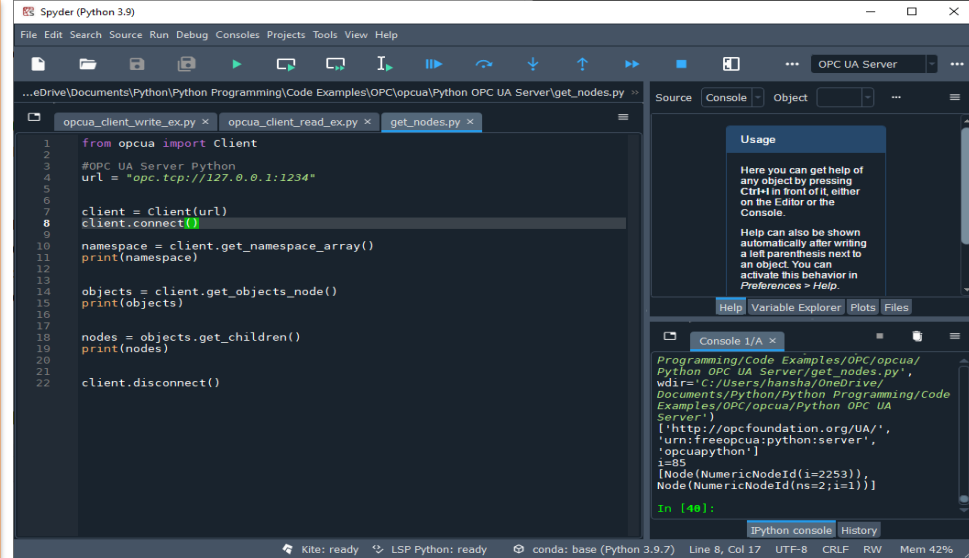
client = Client(url)
client.connect()

namespace = client.get_namespace_array()
print(namespace)

objects = client.get_objects_node()
print(objects)

nodes = objects.get_children()
print(nodes)

client.disconnect()
```



The screenshot shows the Spyder Python IDE interface. The main editor displays the Python code from the previous block. The console window at the bottom shows the output of the code execution:

```
Python console
In [40]:
Out[40]:
[[NamespaceArray(['http://opcfoundation.org/UA/', 'urn:freeopcua:python:server', 'opcua:python'])],
Node(NumericNodeId(i=2253)),
Node(NumericNodeId(ns=2;i=1))]
```

The console also shows a 'Usage' message and a 'Help' message. The status bar at the bottom indicates 'Kite: ready', 'LSP Python: ready', 'conda: base (Python 3.9.7)', 'Line 8, Col 17', 'UTF-8', 'CRLF', 'RW', and 'Mem 42%'.

This gives:  
NodeId = "ns=2;i=2"

# OPC UA Client Write Example

```
from opcua import Client

#OPC UA Server Python
url = "opc.tcp://127.0.0.1:1234"
nodeId = "ns=2;i=2"

client = Client(url)
client.connect()

node = client.get_node(nodeId)
value = float(20)
node.set_data_value(value)

client.disconnect()
```

# Write - Alternative Solution

```
from opcua import Client

#OPC UA Server Python
url = "opc.tcp://127.0.0.1:1234"

client = Client(url)
client.connect()

root = client.get_root_node()
nodeId = root.get_child(["0:Objects", "2:Sensors", "2:Temperature"])
print("NodeId:", nodeId)

node = client.get_node(nodeId)
value = float(20)
node.set_data_value(value)

client.disconnect()
```

# OPC UA Client Read Example

```
from opcua import Client

#OPC UA Server Python
url = "opc.tcp://127.0.0.1:1234"
nodeId = "ns=2;i=2"

client = Client(url)
client.connect()

node = client.get_node(nodeId)
value = node.get_value()
print(value)

client.disconnect()
```

# Read - Alternative Solution

```
from opcua import Client

#OPC UA Server Python
url = "opc.tcp://127.0.0.1:1234"

client = Client(url)
client.connect()

root = client.get_root_node()
nodeId = root.get_child(["0:Objects", "2:Sensors", "2:Temperature"])
print("NodeId:", nodeId)

node = client.get_node(nodeId)
value = node.get_value()
print("Value[°C]:", value)

client.disconnect()
```



# Testing OPC Server and Clients

## OPC UA Server

```
Thonny - C:\Users\hansha\OneDrive\Documents\Python\Python Programming\Code Examples\OPC\opcua\Python OPC UA Server\opcua_server_ex.py @ 22:27
File Edit View Run Tools Help

opcua_server_ex.py x
1 from opcua import Server
2 import time
3 from datetime import datetime
4
5 server = Server()
6 server_url = "opc.tcp://127.0.0.1:1234"
7 server.set_endpoint(server_url)
8
9 name = "opcuapython"
10 namespace = server.register_namespace(name)
11 node = server.get_objects_node()
12 param = node.add_object(namespace, "Sensors")
13 var = param.add_variable(namespace, "Temperature", 0)
14 var.set_writable()
15
16 server.start()
17 print("OPC UA Server Started")
18 print("Press Ctrl-C to Stop Program")
19
20
Shell x
Current Value: 20.0
OPC UA Server Running 13:39:06
Current Value: 20.0
OPC UA Server Running 13:39:07
Current Value: 20.0
OPC UA Server Running 13:39:08
Current Value: 20.0
OPC UA Server Running 13:39:09
Current Value: 20.0
Local Python
```

## OPC UA Clients

```
Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help

...ents\Python\Python Programming\Code Examples\OPC\opcua\Python OPC UA Server\opcua_client_write_ex.py

opcua_client_write_ex.py x  opcua_client_read_ex.py x
1 from opcua import Client
2
3 #OPC UA Server Python
4 url = "opc.tcp://127.0.0.1:1234"
5 nodeId = "ns=2;i=2"
6
7 client = Client(url)
8 client.connect()
9
10 node = client.get_node(nodeId)
11 value = float(20)
12 node.set_data_value(value)
13
14 client.disconnect()

Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial

Help Variable Explorer Plots Files

Console 1/A x
Programming\Code Examples\OPC\opcua\Python OPC UA Server\opcua_client_read_ex.py', wdir='C:/Users/hansha/OneDrive/Documents/Python/Python Programming/Code Examples/OPC/opcua/Python OPC UA Server/'
20.0
In [20]: runfile('C:/Users/hansha/OneDrive/Documents/Python/Python Programming/Code Examples/OPC/opcua/Python OPC UA Server/opcua_client_write_ex.py', wdir='C:/Users/hansha/OneDrive/Documents/Python/Python Programming/Code Examples/OPC/opcua/Python OPC UA Server/')
In [29]:
IPython console History
Kite: ready LSP Python: ready conda: base (Python 3.9.7) Line 4, Col 32 ASCII CRLF RW Mem 42%
```

# OPC UA Python Client GUI

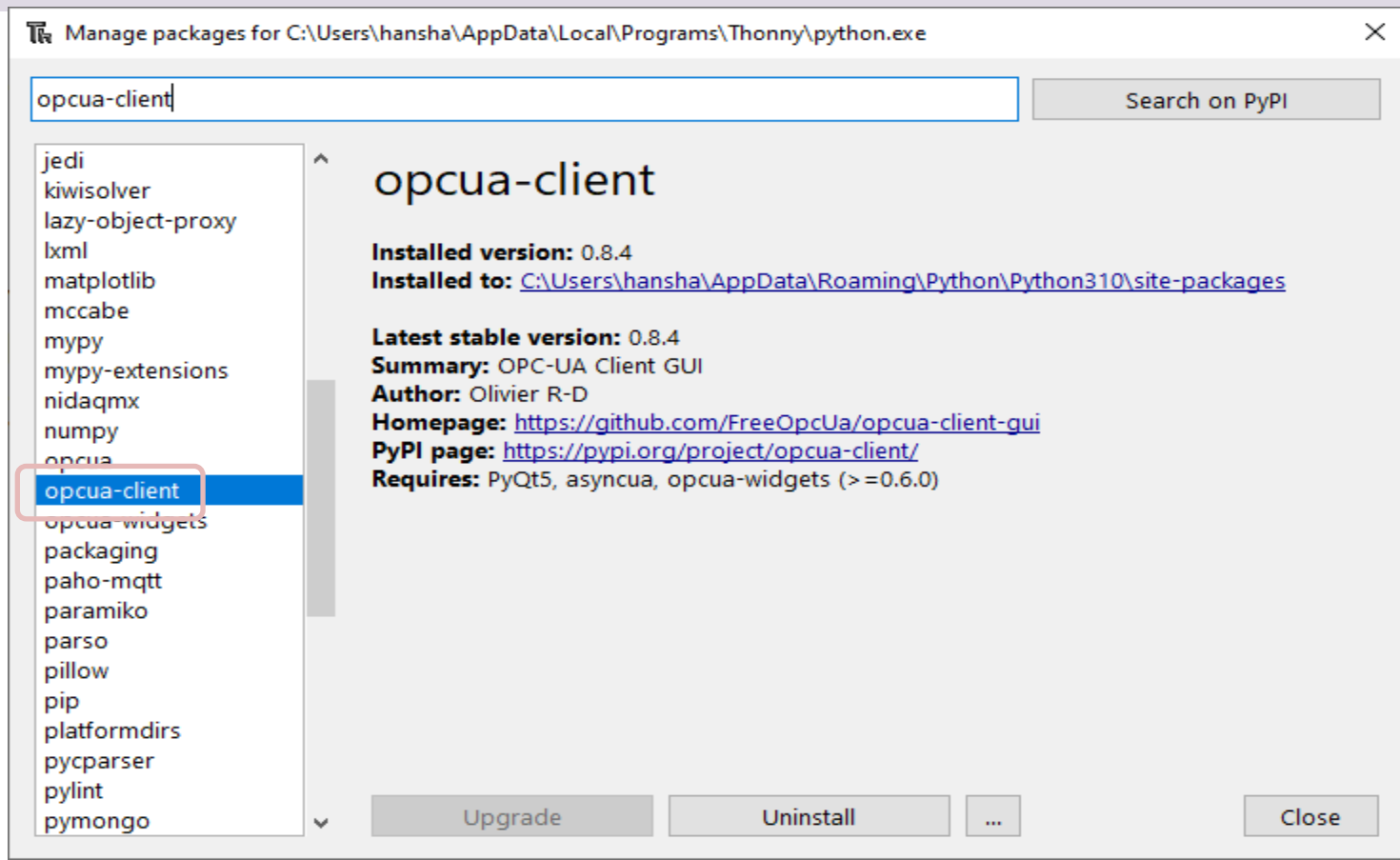
Hans-Petter Halvorsen



# Installation

- OPC UA Python Client GUI
- <https://github.com/FreeOpcUa/opcua-client-gui>

# Installation using Thonny





Manage packages for C:\Users\hansha\AppData\Local\Programs\Thonny\python.exe



pyqtgraph

Search on PyPI

<INSTALL>

adafruit-board-toolkit  
aiofiles  
aiosqlite  
astroid  
asttokens  
asyncua  
bcrypt  
bitstring  
cffi  
colorama  
contourpy  
cryptography  
cyclr  
deprecation  
dill  
dnspython  
docutils  
ecdsa  
esptool  
fonttools  
guizero  
isort

## pyqtgraph

**Latest stable version:** 0.13.3

**Summary:** Scientific Graphics and GUI Library for Python

**Author:** Luke Campagnola

**Homepage:** <http://www.pyqtgraph.org>

**PyPI page:** <https://pypi.org/project/pyqtgraph/>

**Requires:** numpy (>=1.20.0)

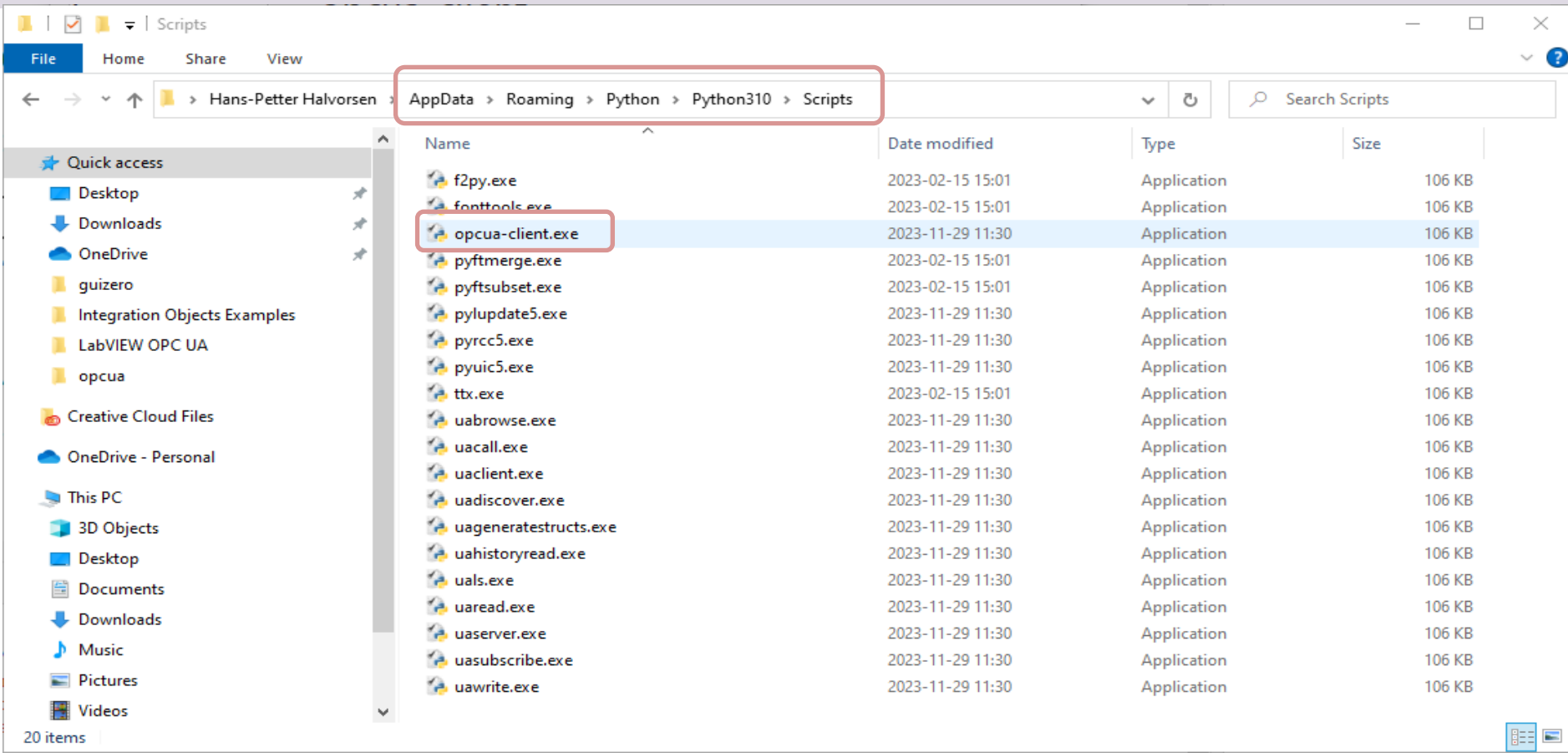
Install

...

Close

You may also need to install “pyqtgraph”

# Location of “opcua-client.exe”



# OPC UA Python Client GUI

The screenshot displays the FreeOpcUa Client application window. The title bar reads "FreeOpcUa Client". Below the title bar are "Actions" and "Settings" menus. The main address bar shows the connection path: `opc.tcp://xps15hph:62640/IntegrationObjects/ServerSimulator`. To the right of the address bar are buttons for "Connect options", "Connect", and "Disconnect".

The main content area is divided into three panes:

- Left Pane (Tree View):** Displays a hierarchical tree of OPC UA objects. The columns are "DisplayName", "BrowseName", and "NodeId". The tree is expanded to show "Tag11" through "Tag9", "Methods", "Real Time Data", "Types", and "Views".
- Right Pane (Attributes):** Displays the attributes of the selected object (Tag7). The columns are "Attribute", "Value", and "DataType". The attributes listed are: AccessLevel (CurrentRead, ...), ArrayDimension (None), BrowseName (0:Tag7), DataType (Double), Description (None), DisplayName (LocalizedText(LocalizedText)), Historizing (True), MinimumSamplingInterval (1.0), and NodeClass (2). A "Refresh" button is located below the table.
- Bottom Pane (Graph):** Displays a graph of the selected object's data. The "Number of Points" is set to 30 and the "Intervall [s]" is set to 5. The graph shows a single data point at approximately 0.1 seconds with a value of 0.2. The x-axis ranges from 0.1 to 0.9, and the y-axis ranges from 0.2 to 0.8. Below the graph are tabs for "Events", "Subscriptions", "References", and "Graph".

The status bar at the bottom of the window displays the following log messages:

```
uaclient.uaclient - INFO - Connecting to opc.tcp://xps15hph:62640/IntegrationObjects/ServerSimulator with parameters None, None, None, None')
asynqua.client.client - WARNING - Deprecated since spec 1.04, call load_data_type_definitions')
asynqua.client.client - WARNING - Deprecated since spec 1.04, call load_data_type_definitions')
```

# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

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

