

Small guide to demonstrate how to communicate with the 2245 sound level meter from within Python.

Introduction

Given that the B&K WebXi protocol is based on HTTP, it will be possible to communicate with the SLM from different platforms, and to use a variety of programming languages to develop your software.

For writing simple programs to interact with the SLM, scripting is an obvious choice. Python is a good candidate, because Python is supported on many platforms and because Python is good at illustrating the basic principles of how to use B&K WebXi.

Setting up Python

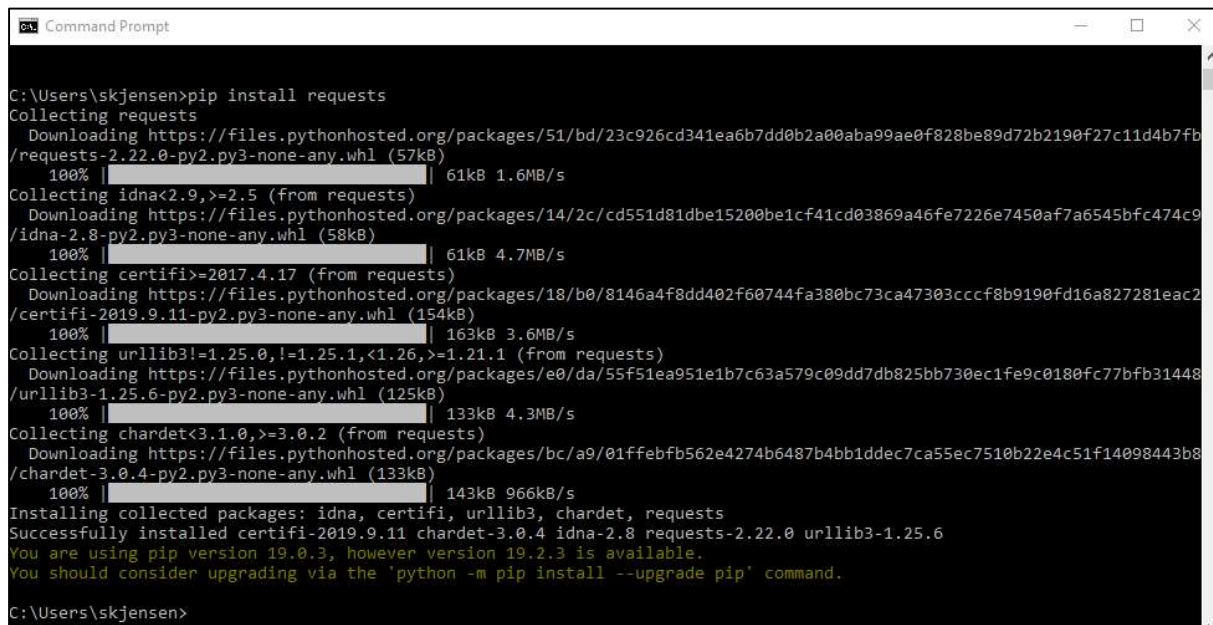
Note: If you already have a running Python environment, you may of course skip this installation.

Download and install the latest Python version from <https://www.python.org/downloads:>

- Select the standard installation
- Check mark the “Add Python to PATH” *)

**) This is just for convenience. You may choose not to, and then remember to add the path to the python commands yourself.*

Once Python has been installed successfully, it is recommended to install the “requests” library for Python. The requests library is a smart way of making HTTP requests in Python, as it abstracts the complexities and makes it is easy to store and inspect the replies. The “requests” library is installed using the ‘pip’ command from the Windows ‘CMD’ command line “Pip install requests”.

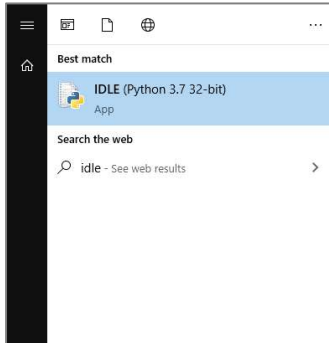


```
C:\Users\skjensen>pip install requests
Collecting requests
  Downloading https://files.pythonhosted.org/packages/51/bd/23c926cd341ea6b7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb
/requests-2.22.0-py2.py3-none-any.whl (57kB)
    100% |#####| 61kB 1.6MB/s
Collecting idna<2.9,>=2.5 (from requests)
  Downloading https://files.pythonhosted.org/packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9
/idna-2.8-py2.py3-none-any.whl (58kB)
    100% |#####| 61kB 4.7MB/s
Collecting certifi>=2017.4.17 (from requests)
  Downloading https://files.pythonhosted.org/packages/18/b0/8146a4f8dd402f60744fa380bc73ca47303ccccf8b9190fd16a827281eac2
/certifi-2019.9.11-py2.py3-none-any.whl (154kB)
    100% |#####| 163kB 3.6MB/s
Collecting urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1 (from requests)
  Downloading https://files.pythonhosted.org/packages/e0/da/55f51ea951e1b7c63a579c09dd7db825bb730ec1fe9c0180fc77bfb31448
/urllib3-1.25.6-py2.py3-none-any.whl (125kB)
    100% |#####| 133kB 4.3MB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests)
  Downloading https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8
/chardet-3.0.4-py2.py3-none-any.whl (133kB)
    100% |#####| 143kB 966kB/s
Installing collected packages: idna, certifi, urllib3, chardet, requests
Successfully installed certifi-2019.9.11 chardet-3.0.4 idna-2.8 requests-2.22.0 urllib3-1.25.6
You are using pip version 19.0.3, however version 19.2.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
C:\Users\skjensen>
```

Wait for the installation of the ‘requests’ library to complete.

Using Get and PUT from within Python, to control the measurement

Executing GET and PUT commands from within “Pythons Integrated Development and Learning Environment” (IDLE) is a quick and easy way to prepare for Python scripting. You can easily try things out and get to know the syntax. You may start IDLE from the Windows start menu:



Below are some examples of how to use GET and PUT from within out the Python IDLE:

A screenshot of the Python 3.7.4 Shell window. The window title is 'Python 3.7.4 Shell'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area contains the following code:

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> # This is a single-line comment. Everything after '#' will be ignored by Python
>>> # You do not need to type the comments; they are there for your information
>>>
>>> # Import 'requests', so that we can access the content in the response in a smart way
>>> import requests
>>> # Issue a GET against the WebXi top node, and store the response in a variable 'myresponse'
>>> myresponse = requests.get("http://10.100.38.87/webxi")
>>> # Investigate the response; JSON formatted
>>> myresponse.json()
{'Applications': {}, 'Sequences': {}, 'Streams': {}, 'Device': {}}
>>> # Issue a GET against the WebXi top node, recursive, and store the response
>>> myresponse = requests.get("http://10.100.38.87/webxi?recursive")
>>> # Investigate the response; JSON formatted
>>> myresponse.json()
Squeezed text (275 lines).
>>> # Note that IDLE will collapse a large number of lines. Double click to expand the result
>>> # Issue a PUT to start a measurement (observe change in the light ring flashing on the SLM)
>>> myresponse = requests.put("http://10.100.38.87/webxi/applications/slm?action=startpause")
>>> # Issue a PUT to pause the measurement (observe change in the light ring flashing on the SLM)
>>> myresponse = requests.put("http://10.100.38.87/webxi/applications/slm?action=startpause")
>>> # Issue a PUT to stop the measurement (observe change in the light ring flashing on the SLM)
>>> myresponse = requests.put("http://10.100.38.87/webxi/applications/slm?action=stop")
>>> |
```

Another example, fetching and writing out the LAF value once per second (note: value is stored in the SLM as dB multiplied by 100):



```
Python 3.7.4 Shell
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> # Import requests
>>> import requests
>>> # Import pretty print, to nicely print out data from the response
>>> import pprint as pp
>>> # import time (to use the sleep method)
>>> import time
>>> # Run program loop 'forever' to fetch LAF from the SLM (or until aborted, eg by ctrl/c)
>>> # Note that the value is stored in the SLM as dB multiplied by 100
>>> while True:
    myresponse = requests.get("http://10.100.38.87/webxi/applications/SLM/Outputs/LAF");
    time.sleep(1)
    pp.pprint(myresponse.json()/100)

33.36
29.59
34.58
36.98
39.27
35.36
91.8
76.3
42.83
38.35
39.26
```

Blow gently into the microphone on the SLM and watch the value change.