

StellarNet

Python Stellarnet_driver3

Documentation v1.1



14390 Carlson Circle
Tampa, FL 33626
+1 (813) 855-8687



CONTENTS

Introduction.....	4
StellarNet Python Driver Installation.....	5
A. Windows.....	5
a. Python 3 Installation & Setup	5
b. Install Packages Using Requirements File	8
c. Install Libusb Filter	9
B. Linux 64 bit (Ubuntu) / MAC (only Intel chip)	12
a. Python 3 Installation & Setup	12
b. Install Libusb Filter	12
c. Install Packages Using Requirements File	12
Getting Started	13
Stellarnet Driver Demo	13
User Guide	15
Tutorial on API Reference call.....	15
API Reference	15
• Class StellarNetError(Exception)	15
o __init__(self, device).....	15
• Class NotFoundError(StellarNetError).....	15
• Class ArgumentError(StellarNetError).....	15
• Class ArgTypeError(ArgumentError)	16
• Class ArgRangeError(ArgumentError)	16
• Class TimeoutError(StellarNetError)	16
• Class Timer.....	16
o __enter__(self)	16
o __exit__(self, *args).....	16
• Class StellarNet(object).....	16
o __init__(self, device).....	16
o __del__(self).....	16
o extrig (self, trigger)	17



○ set_config(self, **kwargs)	17
○ get_config(self)	18
○ get_device_id(self)	19
○ read_spectrum(self)	19
○ get_stored_bytes(self, address)	20
○ get_stored_string(self, address)	20
○ compute_lambda(self, pixel)	21
○ program_FIFO(self)	21
○ _init_config(self)	21
○ print_info(self)	22
○ _set_device_timing(self)	22
○ _read_data(self)	23
○ _smooth_data(self, src)	23
● Other Functions	24
○ _load_firmware(device, filename)	24
○ _set_usb_config(device)	24
○ find_devices()	25
○ _get_params(args)	26
○ select_device(args, return_all=False)	27
○ array_get_spec(chan)	27
○ array_spectrum(spectrometer, wav)	29
○ ext_trig(spectrometer, trigger_val)	30
○ version()	30
License	31



INTRODUCTION

StellarNet USB spectrometer support.

Devices use Cypress Semiconductor CY7C68013A microcontroller (EZ-USB) which initially enumerates as idVendor/idProduct = 0x04B4/0x8613. Documentation for these devices can be found here: <http://www.cypress.com/?rID=38801>.

Folder Contains the following:

- libusb-win32-devel-filter-1.2.6.0
- Python Installation Tutorial-Windows
- stellarnet.hex (HEX File)
- stellarnet_demo.py
 - Demo python file
- requirements.txt
 - TXT file that contains all the libraries needed to run python demo file.
- stellarnet_driverLibs folder that contains library file for different python versions.
 - Extension .pyd is for Windows and extension .so is for Linux/MAC.
 - For example: Cp38 or Cpython-37 indicates python version to use for example: cp38 means python 3.8.
 - A library file containing the stellarnet_driver Python code which is called to and used by the stellarnet_demo.py application.
 - Place this folder in the same folder as the stellarnet_demo.py and at the beginning of the demo file include the following to link the code file with the driver library
 - **from stellarnet_driverLibs import stellarnet_driver3 as sn**

NOTE: Do NOT change any of the file name and file extension.

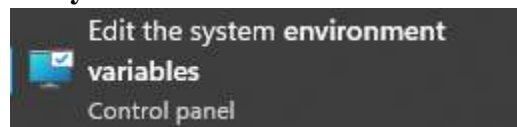


STELLARNET PYTHON DRIVER INSTALLATION

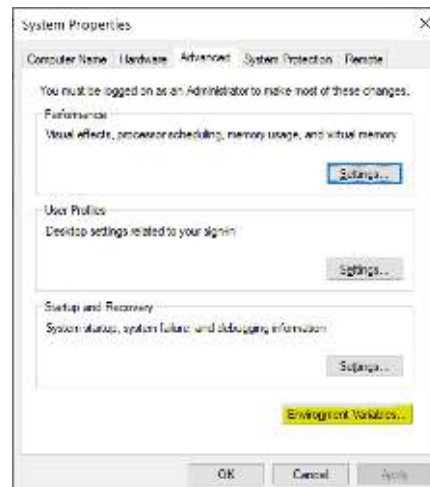
A. Windows

a. Python 3 Installation & Setup

- i. Install Python by navigating to the Python.org [Downloads page](#). A newer version of Python should work, but the driver is being tested with Python 3.8 for this example.
- ii. Add Python to PATH variable
 - A. In the search box on the taskbar at the bottom left hand of your desktop, type “environment variables”, and then select **Edit the system environment variables**

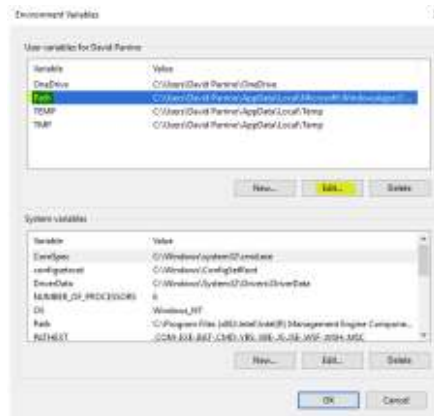


B. Select Environment Variables



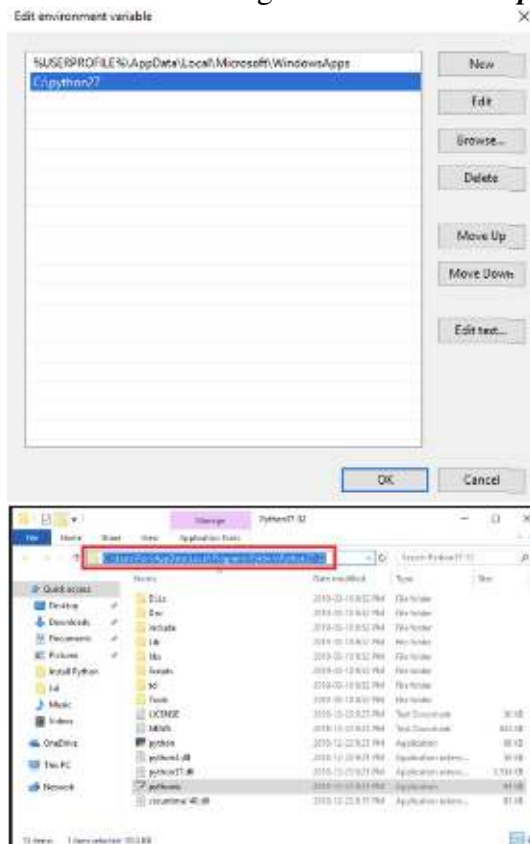


C. Under **user variable for user**, click **Path** and then click **Edit**



D. In the new pop-up window, click **New** and type in the Python application path.

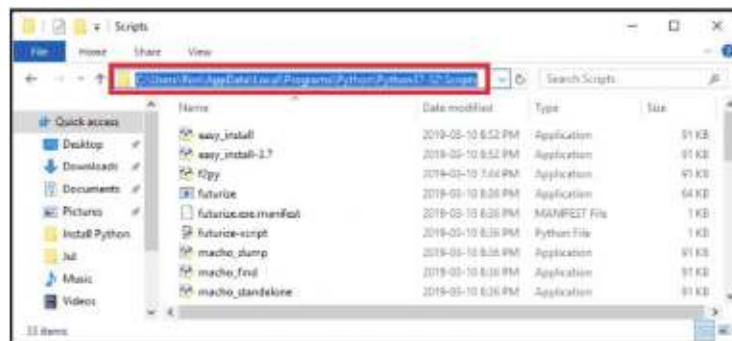
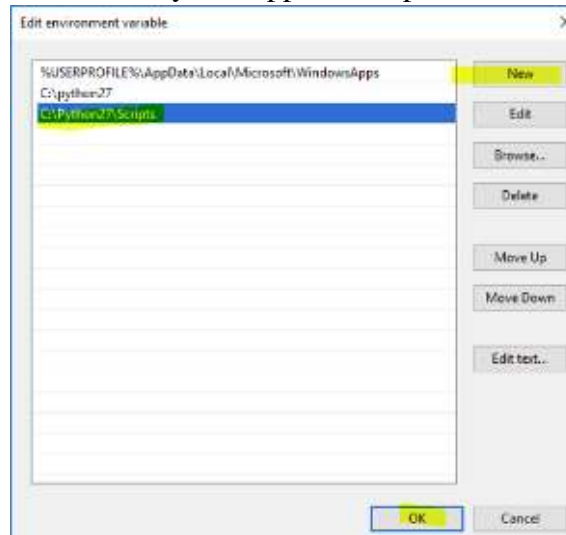
- a. The Python application path is the folder where you originally installed Python. This path can be found by searching for the location of *python.exe*



(An example of the Python application path)

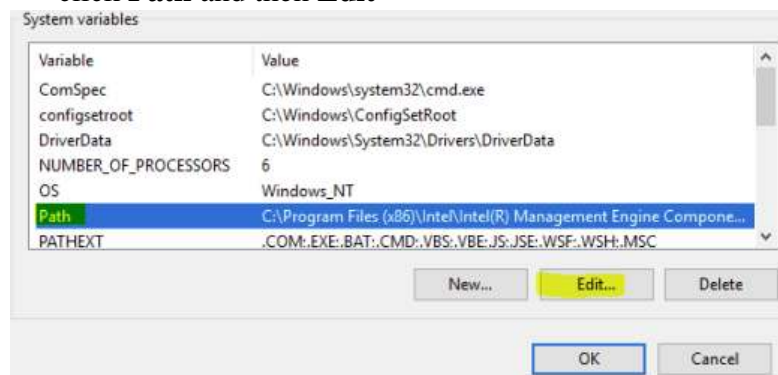


- E. Again, click **New**, and type in the Python Scripts path and click **OK** to close and save the window.
- The Python Scripts folder should be located within the Python application path.



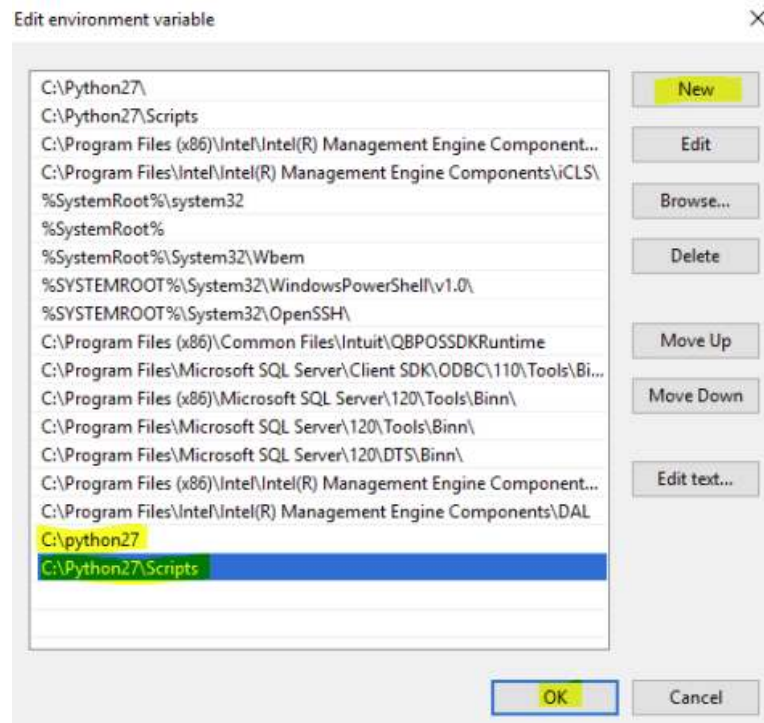
(An example of Python Scripts path)

- F. Under **System variables** in the Environment Variables window, click **Path** and then **Edit**





- G. In the new pop-up window, click **New**, add the same two paths, i.e. the Python application path and Python Scripts path, and click **OK** to close and save the window.



- H. Python now can be used directly from the Command Prompt without having to write its location.

b. Install Packages Using Requirements File

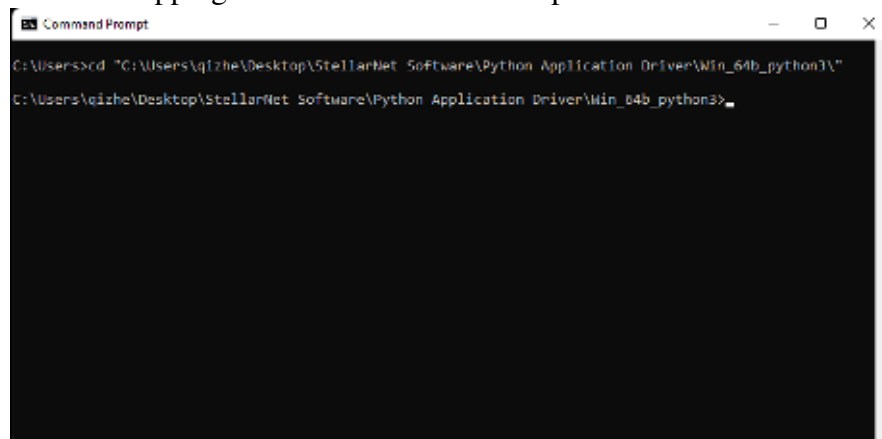
Requirements File, requirements.txt, contains all the python packages that are required to run the driver. Follow the steps below to run and install required package using the requirements file

- i. In the Start Menu, search **Command Prompt**





- ii. In the **Command Prompt**, navigate to the driver folder, which is where the requirements.txt located at, and type “*cd* ” followed by the driver folder path. Note: path autofill can be done by dragging the driver folder and dropping into the Command Prompt window.



- iii. Type “*pip install -r requirements.txt*” and hit enter to install the packages listed in the requirement file.

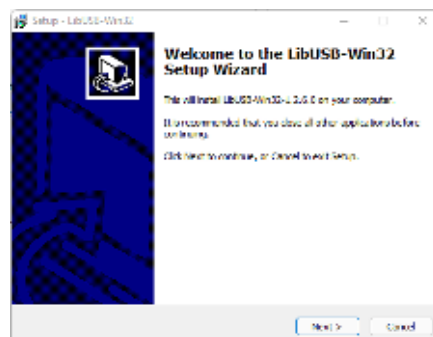
c. Install Libusb Filter

Libusb Filter is needed to redirect the Windows DLL based StellarNet driver from communicating with the spectrometer so that the Python driver will be able to take over. Note, if you’ve never installed the windows driver, you might be able to skip this step. It is often much easier to identify the StellarNet spectrometer if the Windows Driver has already been installed.

- i. Locate and double click *libusb-win32-devel-filter-1.2.6.0* in the folder.

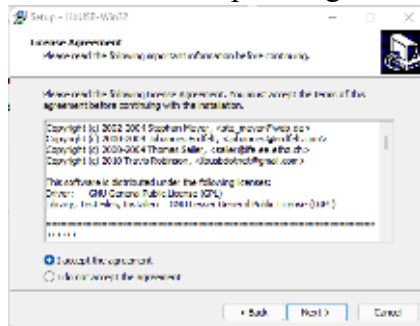


- ii. Click **Next >**

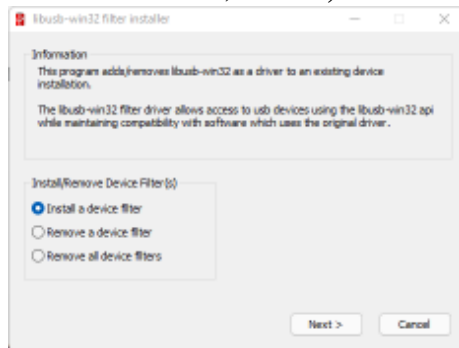




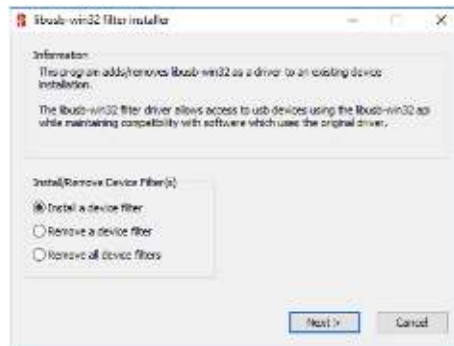
iii. Select “I accept the agreement” and click **Next >**



iv. Click **Next >**, **Install**, and **Finish** until you reach the page below

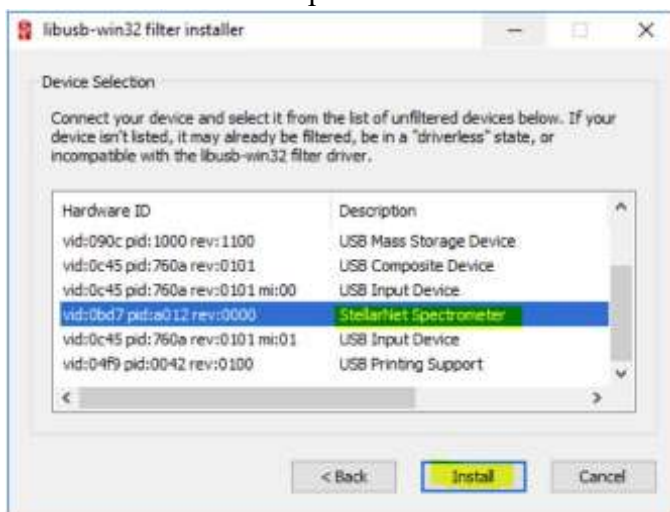


v. Select “Install a device filter” and click **Next >**





- vi. Select “StellarNet Spectrometer” and click **Install**



*Note: If the Windows DLL has not been installed, the Description will not appear as StellarNet Spectrometer.

- vii. The filter should be successfully installed. Click OK to close out the window





B. Linux 64 bit (Ubuntu) / MAC (only Intel chip)

a. Python 3 Installation & Setup

In Linux, Python can be used directly from the Command Prompt without having to write its location.

b. Install Libusb Filter

```
sudo apt-get install libusb-1.0-0-dev
```

c. Install Packages Using Requirements File

The Requirements File, `requirements.txt`, contains all of the python packages that are required to run the driver. Follow the steps below to run and install required packages using the requirements file

- i. In the **Command Prompt**, navigate to the driver folder, where the `requirements.txt` is located, by typing "`cd` " followed by the driver folder path. Note, path autofill can be done by dragging the driver folder and dropping into the Command Prompt window.
- ii. Type "`pip install -r requirements.txt`" and hit enter to install the packages listed in the requirement file.

Note, all dependencies and libraries installation instructions are provided, but for any reason if you get an import error, try to install the unsuccessful library through PIP, or if you have any questions, you can email Support@StellarNet.us with screenshots to illustrate the situation.



GETTING STARTED

Stellarnet Driver Demo

Run stellarnet_demo.py by running *python stellarnet_demo.py* from the command prompt.

```
# import the usb driver - place stellarnet_driverLibs folder in the same folder
# as the stellarnet_demo.py and library can be accessed as follow
from stellarnet_driverLibs import stellarnet_driver3 as sn

# import logging module
import logging
logging.basicConfig(format='%(asctime)s %(message)s')

# Function definition to get data
def getSpectrum(spectrometer, wav):
    logging.warning('requesting spectrum')
    spectrum = sn.array_spectrum(spectrometer, wav)
    logging.warning('recieved spectrum')
    return spectrum

# Function definition to set parameters
def setParam(spectrometer, wav, inttime, scansavg, smooth, xtiming):
    logging.warning('Setting Parameters')
    spectrometer['device'].set_config(int_time=inttime,
    scans_to_avg=scansavg, x_smooth=smooth, x_timing=xtiming)

# Function definition to reset hardware by using ""Destructor. Release device
# resources."" Make sure to call "spectrometer, wav = sn.array_get_spec(0)" to
# init spectrometer again
def reset(spectrometer):
    spectrometer['device'].__del__()

# Function definition to Enable or Disable External Trigger Timeout by Passing
# True or False, If pass True then Timeout function will be disabled. User can
# use this function as timeout enable/disable
def external_trigger(spectrometer, trigger):
    sn.ext_trig(spectrometer, trigger)

# This returns a version number of the compilation date of the driver
version = sn.version()
```



```
#initialize Spectrometer
spectrometer, wav = sn.array_get_spec(0) # 0 for first channel and 1 for
second channel , up to 127 spectrometers

# Device parameters to set
inttime = 50
scansavg = 1
smooth = 0
xtiming = 3

# Get current device parameter
currentParam = spectrometer['device'].get_config()

# Call to Enable or Disable External Trigger by default is Disbale=False
external_trigger(spectrometer,False)

# Check to see if any parameters have changed, id so call setParam
if ((currentParam['int_time'] != inttime) or (currentParam['scans_to_avg'] !=
scansavg) or (currentParam['x_smooth'] != smooth) or (currentParam['x_timing']
!= xtiming)):
    setParam(spectrometer, wav, inttime, scansavg, smooth, xtiming)
# Only call this setParam function on first call to get spectrum and when you
change any parameters i.e inttime, scansavg, smooth.
# Also call getSpectrum twice after this call as first sample of returned data
may not be true for its inttime due to clock interrupt so we throw the first
away and trust the 2nd and subsequent scans.
    for i in range (2): # Call getSpectrum twice and discard first one!
        data=getSpectrum(spectrometer, wav)
else:
    data=getSpectrum(spectrometer, wav)# if no parameters changes just get data
once

# Print data/spect
print(data)

>>> StellarNet Inc USB Driver - Version 2.0 Compiled on 2/2/2022
>>> ('Gain Table From EEPROM:', '4 14 54 94134174 78156187205216 ')
>>> ('Ext trigger is :', False)
>>> 2022-02-16 14:27:23,342 Setting Parameters
>>> 2022-02-16 14:27:23,342 requesting spectrum
>>> 2022-02-16 14:27:23,420 recieved spectrum
>>> 2022-02-16 14:27:23,420 requesting spectrum
>>> 2022-02-16 14:27:23,482 recieved spectrum
>>> [[ 339.13      1499.      ]
>>> [ 339.47636572 1436.      ]
>>> [ 339.8227929  1423.      ]
```



USER GUIDE

Tutorial on API Reference call

References to the code in [Stellarnet Driver Demo](#) section.

- Set spectrometer configuration. See class [StellarNet.set_config](#).
`spectrometer['device'].set_config(int_time=5, scans_to_avg=5, x_smooth=1)`
- To retrieve spectrometer configuration setting. See class [StellarNet.get_config](#).
`spectrometer['device'].get_config()`
- To retrieve spectrometer device id. See class [StellarNet.get_device_id](#).
`spectrometer['device'].get_device_id()`
- Print the device information. See class [StellarNet.print_info](#).
`spectrometer['device'].print_info()`
- To get stored bytes at location 0x08. See class [StellarNet.get_stored_bytes](#).
`spectrometer['device'].get_stored_bytes(0x08)`

API REFERENCE

- [*Class StellarNetError\(Exception\)*](#)
Base class for StellarNet errors. The class inherits the properties and methods from the built-in exception class [Exception](#).
 - `__init__(self, device)`
- [*Class NotFoundError\(StellarNetError\)*](#)
Raised when USB device cannot be found. Inherit the properties and methods from the [StellarNetError](#) class.
- [*Class ArgumentError\(StellarNetError\)*](#)
Raised when argument is in error. Inherit the properties and methods from the [StellarNetError](#) class.



- *Class ArgTypeError(ArgumentError)*
Raised when argument type is incorrect. Inherit the properties and methods from the *ArgumentError* class.
- *Class ArgRangeError(ArgumentError)*
Raised when Argument is out of range. Inherit the properties and methods from the *ArgumentError* class.
- *Class TimeoutError(StellarNetError)*
Raised when device operation times out. Inherit the properties and methods from the *StellarNetError* class.
- *Class Timer*
 - **`__enter__(self)`**
Set the start timer and returns it.
 - **`__exit__(self, *args)`**
Retrieve the end time and compute the start and end time interval.

Parameters: NONE

Return: NONE
- *Class StellarNet(object)*
Represents a StellarNet spectrometer.
 - **`__init__(self, device)`**
Class constructor. Prepares spectrometer device for use.
 - **`__del__(self)`**
Class destructor. Releases spectrometer device resources.



- **extrig (self, trigger)**

Set External Trigger status for the spectrometer device to enable or disable the timeout function.

Parameters:

Name	Data Type	Required/Optional	Description
trigger	Boolean	Required	Ext Trigger status. <i>True</i> for enable and <i>False</i> for disable.

Return: NONE

Example:

```
spectrometer['device'].extrig(True)           //Enable  
spectrometer['device'].extrig(False)          //Disable
```

- **set_config(self, **kwargs)**

Sets the spectrometer device configuration.

Parameters:

Name	Data Type	Required/Optional	Description
int_time	Integer	Optional	The integration time in milliseconds.
x_timing	Integer	Optional	The XTiming rate.
x_smooth	Integer	Optional	The boxcar smoothing window size.
scans_to_avg	Integer	Optional	The number of scans to be averaged together.
temp_comp	Integer	Optional	Temperature compensation (not implemented).

Return: NONE

Example:

```
spectrometer['device'].set_config(int_time = 100, scans_to_avg = 1)
```



- **get_config(self)**
Gets the spectrometer device configuration.

Parameters: NONE

Return:

- A *dict* of the device configuration information
 - **Key:** 'int_time', **Value Type:** *int*
 - **Key:** 'x_timing', **Value Type:** *int*
 - **Key:** 'x_smooth', **Value Type:** *int*
 - **Key:** 'scans_to_avg', **Value Type:** *int*
 - **Key:** 'temp_comp', **Value Type:** *int*
 - **Key:** 'coeffs', **Value Type:** *list*
 - **Key:** 'det_type', **Value Type:** *int*
 - **Key:** 'model', **Value Type:** *str*
 - **Key:** 'device_id', **Value Type:** *str*

Example:

```
config_info = spectrometer['device'].get_config()
print(config_info)

>>> {'int_time': 480, 'x_timing': 1, 'x_smooth': 0, 'scans_to_avg': 1,
'temp_comp': 0, 'coeffs': [0.69267, 0.0001229, 339.13, 0.0], 'det_type': 1,
'model': 'VIS 50', 'device_id': '18011639'}

#Get calebration coefficients info only
coeffs_info = spectrometer['device'].get_config()['coeffs']
print(coeffs_info)

>>> [0.69267, 0.0001229, 339.13, 0.0]
```



- **get_device_id(self)**
Gets the spectrometer device id.

Parameters: NONE

Return: A *str* that represents the device id.

Example:

```
device_id = spectrometer['device'].get_device_id()

print(device_id)

>>> 18011639
```

- **read_spectrum(self)**
Reads and returns a spectrum from the spectrometer.

Parameters: NONE

Return: A *tuple* of short integers

Example:

```
spectrum_info = spectrometer['device'].read_spectrum()

print(spectrum_info)

>>> (1345, 1314, 1312, 1328, 1344, 1331, 1324, 1310, 1320, 1312, 1312,
1309, 1303, 1312, 1312, ..., 1347, 1345, 1347, 1344, 1343, 1340, 1330,
1320, 1331, 1325, 1316, 1317, 1333)
```

Note: Class [StellarNet.set_config\(\)](#) for a description of the parameters that control the operation of the spectrometer or the post-processing of the spectrum.



- **get_stored_bytes(self, address)**

Get stored bytes as bytearray.

Parameters:

Name	Data Type	Required/Optional	Description
address	Integer	Required	The address of the string to get.

Return: A *array.array*, i.e. bytearray, of stored bytes.

Example:

```
bytes_info = spectrometer['device'].get_stored_bytes(0x80)
print(bytes_info)

>>> array('B', [48, 46, 54, 57, 50, 54, 55, 48, 48, 48, 48, 48, 48, 32, 32,
32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 49])
```

- **get_stored_string(self, address)**

Get stored bytes as string.

Parameters:

Name	Data Type	Required/Optional	Description
address	Integer	Required	The address of the string to get.

Return: A *str* of stored bytes.

Example:

```
bytes_str = spectrometer['device'].get_stored_string(____)
print(bytes_str)

>>> 0.6926700000000      1
```



- **compute_lambda(self, pixel)**
Compute lambda from the pixel index.

Parameters:

Name	Data Type	Required/Optional	Description
pixel	Integer	Required	The pixel index on which to perform the computation.

Return: A *float* represents the pixel's wavelength.

Example:

```
lambda_value = spectrometer['device'].compute_lambda(0)
print(lambda_value)
>>>339.13
```

- **program_FIFO(self)**
Set FIFO size (zAP1 with BIG FIFO) and program FIFO.

Parameters: NONE

Return: NONE

Example:

```
spectrometer['device'].program_FIFO()
```

- **_init_config(self)**
Set default configuration.

Parameters: NONE

Return: NONE

Example:

```
spectrometer['device']._init_config()
```



- **print_info(self)**
Print spectrometer device information, i.e. idVendor, idProduct, iManufacturer, and Stored Strings, etc.

Parameters: NONE

Return: NONE

Example:

```
spectrometer['device'].print_info()

>>> --- Device Information
idVendor:   OBD7
idProduct:  A012
iManufacturer: 'StellarNet'
iProduct:   'USB2EPP'
--- Stored Strings:
00 '°x
↑↓ '°x
20 'VIS 50 #18011639f2g1
40 '
60 '°x
80 '0.692670000000      1'
A0 '0.000122900000      0'
C0 '339.130
E0 '0.000000000000      4'
```

- **_set_device_timing(self)**
Send device timing information to the spectrometer device.

Parameters: NONE

Return: NONE

Example:

```
spectrometer['device']._set_device_timing()
```



- **`_read_data(self)`**

Read data from the spectrometer device.

Parameters: NONE

Return: A *tuple* of 16-bit integers from the little-endian data buffer.

Example:

```
data_info = spectrometer['device']._read_data()

print(data_info)

>>> (1279, 1240, 1232, 1246, 1258, 1256, 1266, 1280, 1291, 1290, 1291,
1293, 1281, 1294, 1282, 1279, 1284, 1266, 1265, 1280, 1296, 1334, ...,
1299, 1301, 1284, 1285, 1277, 1277, 1283, 1283, 1284, 1288, 1277, 1279,
1285, 1303)
```

- **`_smooth_data(self, src)`**

Apply boxcar smoothing to data. Smoothing is achieved by smoothing the left end, middle, and right end, where start indexes are inclusive and limit indexes are exclusive for the middle smoothing.

Parameters:

Name	Data Type	Required/Optional	Description
src	tuple	Required	A tuple of read data from the spectrometer device.

Return: A *tuple* of boxcar smoothed data.

Example:

```
smoothed_data =
spectrometer['device']._smooth_data(spectrometer['device']._read_data())

print(smoothed_data)

>>> (1297, 1274, 1271, 1272, 1288, 1289, 1284, 1297, 1299, 1303, 1282,
1274, 1262, 1271, 1287, 1293, 1282, 1282, 1296, 1317, 1335, 1376, ...,
1341, 1356, 1349, 1337, 1333, 1328, 1317, 1310, 1296, 1298, 1298, 1311,
1307, 1329)
```



- **Other Functions**

- **_load_firmware(device, filename)**

Load firmware into device from Intel-hex-formatted file.

Parameters:

Name	Data Type	Required/Optional	Description
device	Stellarnet Object	Required	Stellarnet Class object represent spectrometer device
filename	String	Required	Intel-hex-formatted file name.

Return: NONE

Example:

```
sn._load_firmware(device, 'stellarnet.hex')
```

- **_set_usb_config(device)**

Set USB configuration.

Parameters:

Name	Data Type	Required/Optional	Description
device	StellarNet Object	Required	StellarNet Class object represent spectrometer device

Return: NONE

Example:

```
sn._set_usb_config(device)
```




- **find_devices()**
Find all USB-connected StellarNet devices.

Parameters: NONE

Return: A *tuple* of *StellarNet* objects or raises **NotFoundError** if no devices are found.

Example:

```
sn.find_devices()

>>> 4 14 54 94134174 78156187205216
14
54
94
134
174
78
156
187
205
216
gain setting
changing baseline

print(sn.find_devices())

>>> (<stellarnet_driver3.StellarNet object at 0x000001DC701A9DC0>,)

```



- **_get_params(args)**
Gets the value of the input device setting parameters.

Parameters:

Name	Data Type	Required/Optional	Description
args	Argparse Object	Required	An object of device parameter arguments from argparse .

Return:

- A *dict* of parameter arguments.
 - **Key:** 'device', **Value Type:** *str*
 - **Key:** 'int_time', **Value Type:** *int*
 - **Key:** 'x_timing', **Value Type:** *int*
 - **Key:** 'x_smooth', **Value Type:** *int*
 - **Key:** 'scans_to_avg', **Value Type:** *int*
 - **Key:** 'temp_comp', **Value Type:** *int*
 - **Key:** 'list', **Value Type:** *boolean*
 - **Key:** 'repeats', **Value Type:** *int*

Example:

```
print(sn._get_params(args))

>>> {'int_time': 10, 'x_timing': 20, 'x_smooth': 1, 'scans_to_avg': 2,
'temp_comp': 3}
```



○ **select_device(args, return_all=False)**

If device not found, raise error message and system exit. If device found and return_all = True, return all the devices in a *tuple* of *StellarNet objects*; else return the first *StellarNet* object.

Parameters:

Name	Data Type	Required/Optional	Description
args	Argparse Object	Required	An object of device parameter arguments from argparse .
Return_all	Boolean	Optional; default to False	Return all <i>StellarNet</i> object or not. True for yes and False for no.

Return: If device found and return_all = True, return all the devices in a *tuple* of *StellarNet objects*; else return the first *StellarNet* object.

Example:

```
print(sn.select_device (args))

>>> <stellarnet_driver3.StellarNet object at 0x000001D1A39260A0>
```

○ **array_get_spec(chan)**

Initiate Spectrometer device and setup parameter for acquisition

Parameters:

Name	Data Type	Required/Optional	Description
chan	Integer	Required	Number of channels upto 127 and 0=1 Spectrometer

Return:

- Spectrometer as *dict* with data type *str* as the key:
 - **Key:** 'device', **Value type:** *StellarNet* object
 - **Key:** 'config_id', **Value type:** *int*.
- *numpy.ndarray* for the wavelength data.

Example:

```
#print all return information
print(sn.array_get_spec(0))

>>> {'device': <stellarnet_driver3.StellarNet object at
0x0000023A5CD150A0>, 'config_id': 0}, array([[ 339.13    ],
      [ 339.47636572],
      [ 339.8227929 ],
      ...,
      [1175.87779313],
      [1176.3498241 ],
      [1176.82191653]]))

#print spectrometers information
print(sn.array_get_spec(0)[0])

>>> {'device': <stellarnet_driver3.StellarNet object at
0x0000023A5CD150A0>, 'config_id': 0},

#print spectrometers information
print(sn.array_get_spec(0)[0]['device'].get_device_id())

>>> 18011639
```



- **array_spectrum(spectrometer, wav)**

Call read_spectrum to get spectrum and return as 2D array of Calibrated Wavelength vs Counts.

Parameters:

Name	Data Type	Required/Optional	Description
spectrometer	Dictionary: Key: 'device', Value type: <i>StellarNet</i> object. Key: 'config_id', Value type: <i>int</i> .	Required	Spectrometer device and configuration information.
wav	numpy.ndarray	Required	Wavelength data

Return: A *numpy.ndarray* of spectrum array.

Example:

```
print(sn.array_spectrum(spectrometer, wav))
```

```
>>> [[ 339.13    1439.    ]
      [ 339.47636572 1412.    ]
      [ 339.8227929  1410.    ]
      ...
      [1175.87779313 1410.    ]
      [1176.3498241  1408.    ]
      [1176.82191653 1414.   ]]
```



- **ext_trig(spectrometer, triggerval)**
Enable or disable Ext Trigger for the selected spectrometer device. Timeout function will be disabled if variable *triggerval* is set to True. User can also use this function as timeout enable or disable.

Parameters:

Name	Data Type	Required/Optional	Description
spectrometer	Dictionary: Key: 'device', Value type: <i>StellarNet</i> object. Key: 'config_id', Value type: <i>int</i> .	Required	Spectrometer device and configuration information.
triggerval	Boolean	Required	Status for the Ext Trigger to be set.

Return: NONE

Example:

```
sn.ext_trig(spectrometer, False)
```

- **version()**
Keep track of the version for the StellarNet Inc USB Driver as well as the latest compiled date.

Parameters: NONE

Return: A *str* that contains the version and latest compiled date information.

Example:

```
print(sn.version)  
  
>>>'StellarNet Inc USB Driver - Version 2.0 Compiled on
```



LICENSE

Copyright 2021 StellarNet, Inc.

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.