StellarNet

Python SDK Stellarnet_driver3

Documentation v2.5

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

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

CONTENTS

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

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

INTRODUCTION

StellarNet USB spectrometer support. Devices use Cyprus 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:

- stellarnet_demo.py
  - Demo python file
- stellarnet_driverLibs folder that contains library file for different python versions.
  - Windows_only
    - InstallDriver.exe → Used by the function installDeviceDriver() to install the Driver
      - StellarNet_Python_USB_Driver → Used by the function uninstallDeviceDriver() to remove the Python USB Driver.
    - libusb-win32-devel-filter-1.2.6.0 → A manual install and uninstall can also be done by running this filter executable.
  - Library files containing the compiled stellarnet_driver Python code and it is used by the stellarnet_demo.py application.
    - Compiled Python driver for different operating systems, bit size and Python version.

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

A. Compiled Driver Support

The table shows a list of the available operating systems, bit size, and the Python version for the StellarNet Driver. All of the compiled driver files will be in the "Stellarnet_driverLibs" folder.

Operating Systems	File Format	Python Support
Window (64 bits)	stellarnet_driver3.cp3*-win_amd64.pyd	3.5 to 3.10
Window (32 bits)	stellarnet_driver3.cp3*-win32.pyd	3.5 to 3.10
Linux	stellarnet_driver3.cpython-3*m-x86_64-linux-gnu.so	3.5 to 3.10
Mac	stellarnet_driver3.cpython-3*-darwin.so	3.8 to 3.10
Raspberry pi (32 bits)	stellarnet_driver3.cpython-3*m-arm-linux-gnueabihf.so	3.4 to 3.7
Raspberry pi (64 bits)	stellarnet_driver3.cpython-39-aarch64-linux-gnu.so	3.9

NOTE:

- Do NOT change any of the file names and file extensions.
- You may delete any compiled driver file(s) which are not suitable for your operating system/bit size/Python version to save space. For example, if you are only using the driver for 64 bit Windows OS, then other compiled drivers with the extension of ".so" can be removed as well as "stellarnet_driver3.cp3*-win32.pyd".
- Place this folder in the same folder as the stellarnet_demo.py and at the beginning of the demo script should include the following statement to link the code file with the driver library
    - `from stellarnet_driverLibs import stellarnet_driver3 as sn`

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

STELLARNET PYTHON DRIVER INSTALLATION

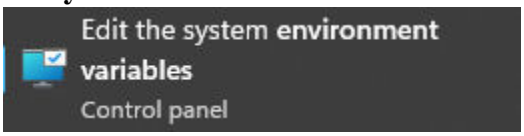
B. Windows

a. Python 3 Installation & Setup

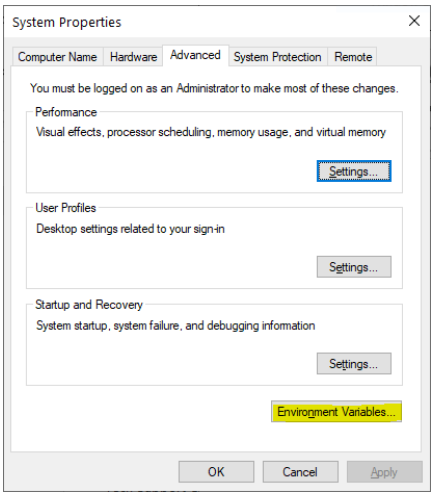
i. Install Python by navigating to the Python.org Downloads page. A newer version of Python should work, but the driver is being tested with Python 3.9 for this example.

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**

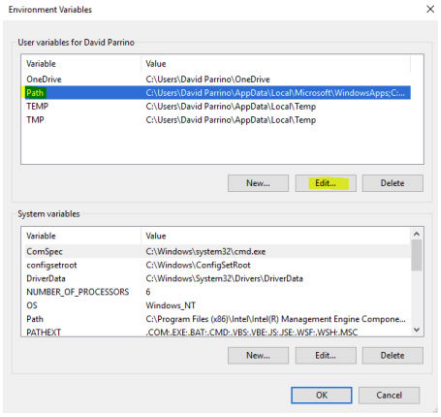


B. Select **Environment Variables**



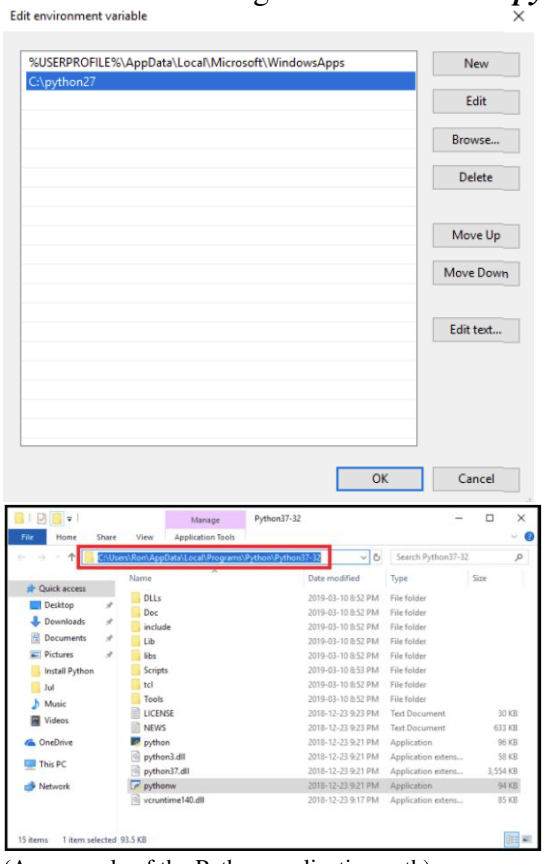
14390 Carlson Circle
Tampa, FL 33626
+1 (813) 855-8687

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.

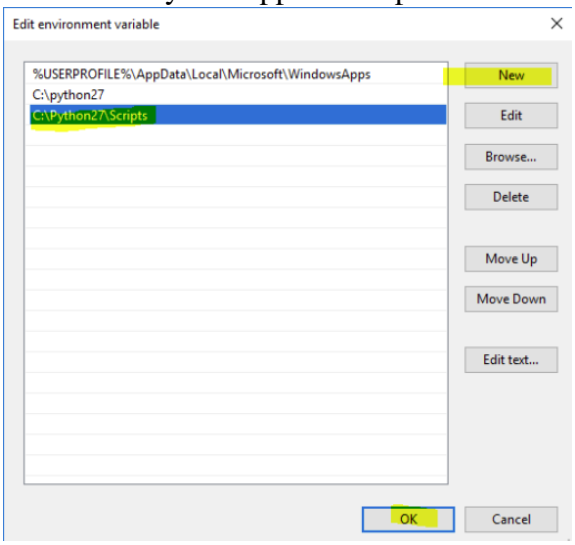
   a. The Python application path is the folder where you originally installed Python. This path can be found by searching for the location of ***python.exe***

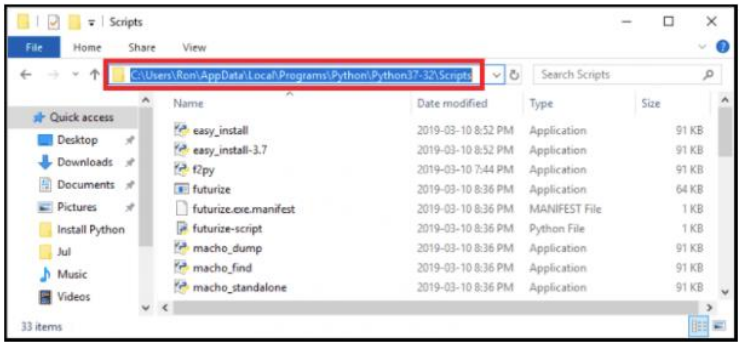


(An example of the Python application path)

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

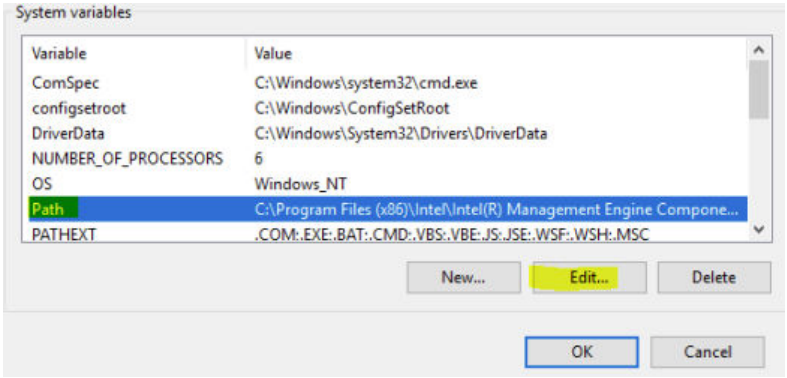
E. Again, click **New**, and type in the Python Scripts path and click **OK** to close and save the window.
    a. The Python Scripts folder should be located within the Python application path.





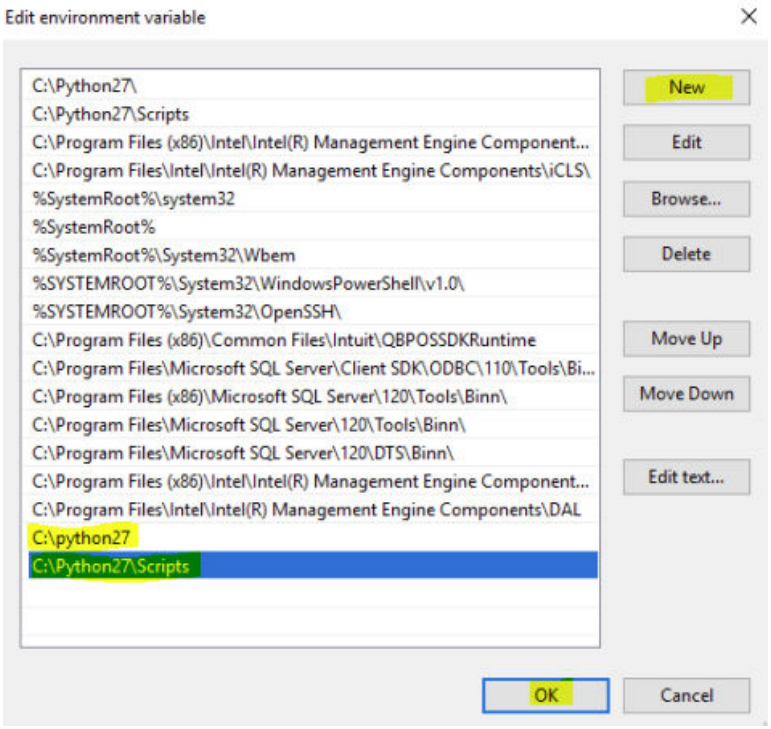
(An example of Python Scripts path)

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



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

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

Python drivers require some packages and libraries. Follow the steps below to install all the dependencies.

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

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

iii. In the terminal, type *"pip install numpy==1.24.2"* to install numpy package

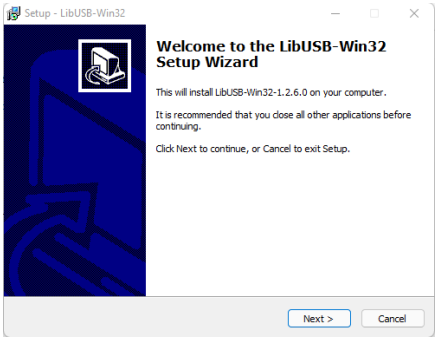
iv. In the terminal, type *"pip install pyusb==1.2.1"* to install pyusb package

c. Install Libusb Filter

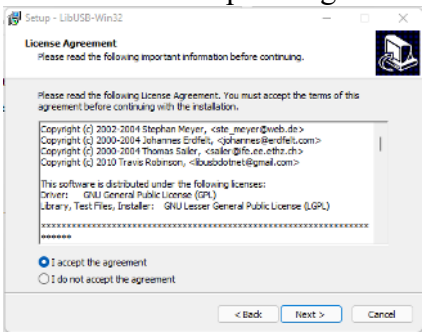
Libusb Filter is needed to redirect the Windows DLL based StellarNet driver from communicating with the spectrometer so that the Python driver will be able to take over. Note, if you've never installed the windows driver, you might be able to skip this step. It is often much easier to identify the StellarNet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

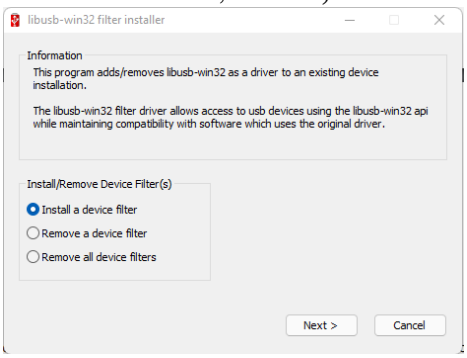


iii. Select "I accept the agreement" and click **Next >**

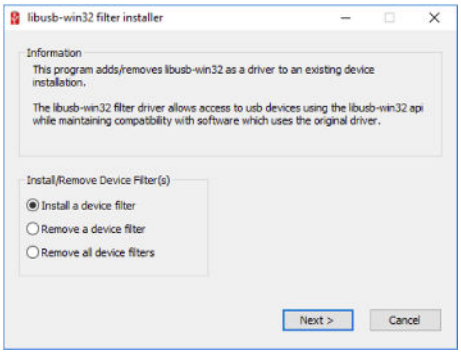


14390 Carlson Circle
Tampa, FL 33626
+1 (813) 855-8687

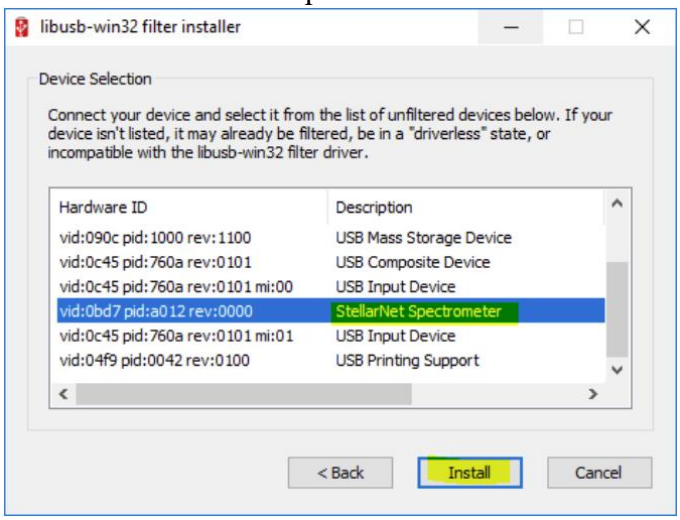
iv. Click **Next >**, **Install,** and **Finish** until you reach the page below



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.

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

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

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

C. 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

Python drivers require some packages and libraries. Follow the steps below to install all the dependencies.

i. In the terminal
ii. In the terminal, type *"pip install numpy==1.24.2"* to install numpy package
iii. In the terminal, type *"pip install pyusb==1.2.1"* to install pyusb package

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.

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

GETTING STARTED

Stellarnet Driver Demo

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

```
try:
    from stellarnet_driverLibs import stellarnet_driver3 as sn
except:
    print("\n\n*********************************** ERROR
*****************************************")
    print("                 Compatible Python Driver DOES NOT EXIST")
    print("       ** See \"stellarnet_driverLibs\" for all compiled Drivers")
    print(" ** See \"Compiled Driver Supports\" in the documentation for all
compiled Drivers ")
    print("*********************************** ERROR
*****************************************\n\n")
    quit()

# For Windows ONLY: Must be run in administrator mode
# Only need to run it one time after switch back from the SpectraWiz.
#sn.installDeviceDriver()

# This resturn a Version number of compilation date of driver
version = sn.version()
print(version)

# Device parameters to set
inttime = 50   # 1-498000 ms
scansavg = 1   # > 1
smooth = 0     # 1-4
xtiming = 3    # 1-4

#init Spectrometer - Get BOTH spectrometer and wavelength
spectrometer, wav = sn.array_get_spec(0) # 0 for first channel and 1 for second
channel , up to 127 spectrometers
"""
# Equivalent to get spectrometer and wav separately:
spectrometer = sn.array_get_spec_only(0)
wav = sn.getSpectrum_X(spectrometer)
"""
```

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


```
print(spectrometer)
sn.ext_trig(spectrometer, True)

# Get device ID
deviceID = sn.getDeviceId(spectrometer)
print('\nMy device ID: ', deviceID)

# Get current device parameter
currentParam = sn.getDeviceParam(spectrometer)

# Call to Enable or Disable External Trigger to by default is Disbale=False ->
with timeout
# Enable or Disable Ext Trigger by Passing True or False, If pass True than
Timeout function will be disable, so user can also use this function as
timeout enable/disbale
sn.ext_trig(spectrometer,True)

# Only call this function on first call to get spectrum or when you want to
change device setting.
# -- Set last parameter to 'True' throw away the first spectrum data because
the data may not be true for its inttime after the update.
# -- Set to 'False' if you want to throw away the first data, however your
next spectrum data might not be valid.
sn.setParam(spectrometer, inttime, scansavg, smooth, xtiming, True)

# Get spectrometer data - Get BOTH X and Y in single return
first_data = sn.array_spectrum(spectrometer, wav) # get specturm for the first
time
"""
# Get Y value ONLY :
first_data = sn.getSpectrum_Y(spectrometer)
"""
print('First data:', first_data )

#=============================================
# Burst FIFO mode: Not recommended with high integration time.
# burst_data_2 = sn.getBurstFifo_Y(spectrometer)
#=============================================

# Release the spectrometer before ends the program
sn.reset(spectrometer)

# For Windows ONLY: Must be run in administrator mode
# sn.uninstallDeviceDriver()
```

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

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()`

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

API REFERENCE

- *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.
    Parameters:

Name	Data Type	Required/Optional	Description
trigger	Boolean	Required	Ext Trigger status. *True* for no timeout and *False* for with timeout.

    Return: NONE

    Example:

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

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

- **set_config(self, **kwargs)**
  Set and configure the the spectrometer device, includes the integration time, smoothing, scans to average, X timing and temperature compensation.

  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.

  Return: NONE

  Example:

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

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

- **get_config(self)**
  Gets the current spectrometer device configuration, includes the integration time, smoothing, scans to average, X timing and temperature compensation.

  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]
```

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

- **get_device_id(self)**
  Gets the spectrometer device ID in string.

  Parameters: NONE

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

  Example:

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

print(device_id)

>>> 18011639
```

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

- **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.

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

- **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
```

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

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

  Parameters: NONE

  Return: NONE

  Example:

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

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

- **_init_config(self)**
  Set default configuration for the spectrometer.

  Parameters: NONE

  Return: NONE

  Example:

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

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

- **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:      0BD7
idProduct:     A012
iManufacturer: 'StellarNet'
iProduct:      'USB2EPP'
--- Stored Strings:
00 '°×
↕ ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ'
20 'VIS 50 #18011639f2g1          '
40 '                              '
60 'ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ'
80 '0.692670000000              1'
A0 '0.000122900000              0'
C0 '339.130                     '
E0 '0.000000000000              4'
```

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

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

  Parameters: NONE

  Return: NONE

  Example:

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

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

- **_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)
```

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

- **_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)
```

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

- **_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)
```

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

- **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>,)
```

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

- **_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}
```

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

- **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 *StellarNet* 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>
```

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

- **array_get_spec(chan)**
  Initiate Spectrometer device and setup parameter for acquisition. Return the spectrometer object and the spectrometer wavelength.

  Parameters:

Name	Data Type	Required/Optional	Description
chan	Integer	Required	Number of channels up to 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
```

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

- **array_get_spec_only (chan)**
  Initiate Spectrometer device and setup parameter for acquisition. Return the spectrometer object only. Exclude spectrum wavelength – called 'getSpectrum_X' to get the spectrum wavelength.

  Parameters:

Name	Data Type	Required/Optional	Description
chan	Integer	Required	Number of channels up to 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*.

  Example:

```
spectrometer = sn.array_get_spec_only(0)
wav = sn.getSpectrum_X(spectrometer)
```

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

- **array_spectrum(spectrometer, wav)**
  Get spectrum data from the corresponding spectrometer. Return a 2D array of Calibrated Wavelength (X) and the spectrum Counts (Y).
  Parameters:

Name	Data Type	Required/Optional	Description
spectrometer	Dictionary: **Key**: 'device', **Value type**: *StellarNet* object. **Key**: 'config_id', **Value type**: *int*.	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.       ]]
```

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

- **ext_trig(spectrometer, triggerval)**
  Enable or disable Ext Trigger for the selected spectrometer device. Timeout function will be disable 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**: *StellarNet* object. **Key**: 'config_id', **Value type**: *int*.	Required	Spectrometer device and configuration information.
triggerval	Boolean	Required	Status for the Ext Trigger to be set. *True* of no timeout and *False* for with timeout.

  Return: NONE

  Example:

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

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

- **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
2/2/2022'
```

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

- **getSpectrum_X(spectrometer)**
  Get the wavelength of the spectrometer. Exclude Y axis of the spectrum data – called 'getSpectrum_Y' to get the spectrum data.

  Parameters:

Name	Data Type	Required/Optional	Description
spectrometer	Dictionary: **Key**: 'device', **Value type**: *StellarNet* object. **Key**: 'config_id', **Value type**: *int*.	Required	Spectrometer device and configuration information.

  Return: A *list* of the X (wavelength) axis of spectrum data

  Example:

```
spectrometer = sn.array_get_spec_only(0)
wav = sn.getSpectrum_X(spectrometer)
```

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

- **getSpectrum_Y(spectrometer)**
  Get spectrum data from the spectrometer. Exclude spectrum wavelength – called 'getSpectrum_X' to get the spectrum wavelength.

  Parameters:

Name	Data Type	Required/Optional	Description
spectrometer	Dictionary: **Key**: 'device', **Value type**: *StellarNet* object. **Key**: 'config_id', **Value type**: *int*.	Required	Spectrometer device and configuration information.

  Return: A *list* of the Y axis of spectrum data

  Example:

```
spectrometer = sn.array_get_spec_only(0)
spec_y = sn.getSpectrum_Y(spectrometer)
```

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

- **getDeviceId(spectrometer)**
  Returns the spectrometer device ID.

  Parameters:

Name	Data Type	Required/Optional	Description
spectrometer	Dictionary: **Key**: 'device', **Value type**: *StellarNet* object. **Key**: 'config_id', **Value type**: *int*.	Required	Spectrometer device and configuration information.

  Return: A *str* of device ID.

  Example:

```
print(sn.getDeviceParam(spectrometer))

>>>22072014
```

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

- **getDeviceParam(spectrometer)**

  Returns the current spectrometer device parameters.

  Parameters:

Name	Data Type	Required/Optional	Description
spectrometer	Dictionary: **Key**: 'device', **Value type**: *StellarNet* object. **Key**: 'config_id', **Value type**: *int*.	Required	Spectrometer device and configuration information.

  Return: A *dictionary* of spectrometer device parameters.

  Example:

```
print(sn.getDeviceParam(spectrometer))

>>>{'int_time': 50, 'x_timing': 3, 'x_smooth': 0,
'scans_to_avg': 1, 'temp_comp': False, 'coeffs': [0.69267,
0.0001229, 339.13, 0.0], 'det_type': 1, 'model': 'BW ZAP',
'device_id': '22072014'}
```

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

- **setParam(spectrometer, inttime, scansavg, smooth, xtiming, clear =True)**
  Configure the spectrometer device parameters as well as clear (if clear is set to True) the first spectrum data after the parameter setting to receive valid spectrum data.

  Parameters:

Name	Data Type	Required/Optional	Description
spectrometer	Dictionary: **Key**: 'device', **Value type**: *StellarNet* object. **Key**: 'config_id', **Value type**: *int*.	Required	Spectrometer device and configuration information.
inttime	Integer	Required	Integration time
scansavg	Integer	Required	Number of scans to average
smooth	Integer	Required	Degree of smoothing
xtiming	Integer	Required	X timing
clear	Boolean	Required	Throw away for spectrum data after the device update. *It is recommended to set it to *True* if integration time and/or Xtiming is changed.

  Return: None

  Example:

```
# Integration time = 100, Scans to average = 4, smooth = 2,
Xtiming = 3
sn.setParam(spectrometer, 100,4,2,1)
```

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

- **setTempComp(spectrometer, temp_comp)**
  Set temperature compensation to reflect on the returned on the spectrum data. There are 15 “optically black” pixels on StellarNet’s SONY ILX-511b CMOS detectors which are not hit by light during an acquisition. They provide a continuous measurement of the average dark spectrum and can be used to adjust for baseline drift during an experiment. In other words, this feature compensates for changes in the baseline due to temperature.

  Parameters:

Name	Data Type	Required/Optional	Description
spectrometer	Dictionary: **Key**: ‘device’, **Value type**: *StellarNet* object. **Key**: ‘config_id’, **Value type**: *int*.	Required	Spectrometer device and configuration information.
temp_comp	Boolean	Required	Turn on or off the temperature compensation calculation

  Return: None

  Example:

```
sn.setTempComp(spectrometer, True) # Enable temperature
compensation

sn.setTempComp(spectrometer, False) # Disable temperature
compensation
```

  Note:

  This feature is not useful for Hamamatsu CMOS detectors as these do not have dark pixels.

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

- **getBurstFifo_Y(spectrometer)**
  Burst Mode requires a zAP1 board upgrade with a large FIFO. When operating in Burst Mode, 128 consecutive spectra are collected into the large FIFO at the configured integration time and then sent as one packet over USB. This feature is recommended for integration times shorter than 30-50ms as real time acquisition is compromised by the data transfer polling rate of USB hardware. The polling rate is approximately 30Hz but can vary depending on hardware and operating system conditions. Burst mode is NOT recommended for longer integration times. Scans to average will not be applied for any Burst Mode capture.

  Parameters:

Name	Data Type	Required/Optional	Description
spectrometer	Dictionary: **Key**: 'device', **Value type**: *StellarNet* object. **Key**: 'config_id', **Value type**: *int*.	Required	Spectrometer device and configuration information.

  Return: A *list* of *tuples* spectrum data. The list will contain a total of 128 tuples and each tuple stored the retrieved spectrum data.

  Example:

```
Burst_data = sn.getBurstFifo_Y(spectrometer)
```

  Note:

  Burst Mode is only available in zAP1 boards.

  Burst mode is not available in standard board configurations.

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

- **reset(spectrometer)**
  Release the spectrometer object. It is recommended at the end of the program to release the object.

  Parameters:

Name	Data Type	Required/Optional	Description
spectrometer	Dictionary: **Key**: 'device', **Value type**: *StellarNet* object. **Key**: 'config_id', **Value type**: *int*.	Required	Spectrometer device and configuration information.

  Return: None

  Example:

```
sn.reset()
```

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

- **installDeviceDriver ()**
  This function only applies to the Windows Operating System to avoid a conflict with the SpectraWiz software. This function only needs to be run ONCE when you run the script for the first time. When you are ready to switch back to the SpectraWiz, make sure you run uninstallDeviceDriver() to uninstall the device driver.

  - After this function is executed, a popup window titled "Device Driver Installation Wizard' will show up. Click "Next" to install it, and then click "Finish".

  Note:

  - To utilize this function, the script MUST be running in Administrator mode.
  - This function must be called at the beginning of the script to enable the device driver.
  - If your computer has some type of anti-virus software, this function might be blocked, either allow the access from the anti-virus software or a manual install of the device driver (or Filter Wizard) will be needed.

  Parameters: NONE

  Return: None

  Example:

```
sn.installDeviceDriver()
```

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

- **uninstallDeviceDriver()**
  This function is for the Windows Operating System with the installed SpectraWiz software only. This function will avoid the conflict of the USB utility for the programming script with the SpectraWiz, so uninstalling the device driver using this function will allow the use of the SpectraWiz Software. This function should be called whenever you are ready to use the SpectraWiz software after the programming.

  Note:

  - To utilize this function, the script MUST be running in the Administrator mode.

  Parameters: NONE

  Return: NONE

  Example:

```
sn.uninstallDeviceDriver()
>>>Microsoft PnP Utility

Driver package uninstalled.
Driver package deleted successfully.
done
```

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

VERSION UPDATE

A. Version 2.5

- ✓ Hot pixel mapping
- ✓ Easy access to device Count, device ID, device parameters
- ✓ Get wavelength (X) and Spectrum (Y) separately as a single list
- ✓ Set and retrieve device parameters
- ✓ T3 detector support
- ✓ Burst mode for CCD and CMOS
- ✓ Auto-driver install and uninstall on Windows
- ✓ Temperature compensation
- ✓ Support for more Operating Systems, bitness, and Python Versions

LICENSE