biosignal acquisition tool-kit for advanced research applications

# MATLAB® compatibility with biosignalsplux Python™ API

This document serves to provide guidance on using biosignalsplux with MATLAB® using the official biosignalsplux PythonTM API on Windows operating systems.

The information contained in this document has been carefully checked and we made every effort to ensure its quality. PLUX reserves the right to make changes and improvements to this manual and products referenced at any time without notice.

**PLUX Wireless Biosignals S.A.**
email: plux@plux.info
web: http://www.plux.info

Headquarters
Zona Industrial das Corredouras, Lt. 14 – 1°
2630-369 Arruda dos Vinhos
Portugal
tel.: +351 263 978 572
fax: +351 263 978 902

Lisbon Office
Av. 5 de Outubro, n° 79 – 8°
1050-059 Lisboa
Portugal
tel.: +351 211 956 542
fax: +351 211 956 546

# DISCLAIMER

**biosignalsplux products** are intended for use in **life science education and research applications only**; they are **not medical devices**, nor are they intended for medical diagnosis, cure, mitigation, treatment or prevention of disease and is provided to you “as is”.

We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost revenues, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted, even if advised of the possibility of such damages.

# TABLE OF CONTENTS

# 1 Introduction

This document serves to provide guidance on using biosignalsplux with MATLAB® (MATLAB) using the official biosignalsplux Python™ (Python) API.

The PLUX Python API brings to Python applications all the functionality of PLUX devices. The `BaseDev` class (or any of its derived classes) encapsulates a connection to a PLUX device. Each `BaseDev`-derived class provides methods to access the specific functionality of certain PLUX devices.

This API is not thread-safe since its methods cannot be called simultaneously by different threads. The only exception to this rule is the `BaseDev.interrupt( )` method. This API is implemented as a wrapper around the PLUX C++ API, with similar classes and methods, and it is delivered as a binary **.PYD** file compiled for Python 2.7.

It is recommended to review other official biosignalsplux documentation when following this document.

biosignalsplux user manual:
http://biosignalsplux.com/downloads/biosignalsplux_User_Manual_v.1.0.pdf

Datasheets:
http://biosignalsplux.com/en/learn/documentation

# 2 Requirement & Setup

Please follow the steps below to prepare your system to use the biosignalsplux Python API in MATLAB. Note, that the API itself is only supported for Windows operating system and will not work on other systems.

## 2.1 Python Requirements

Unlike most Unix systems, Windows does not contain a system supported Python installation, therefore, Python must be downloaded and installed manually by the user.

Download a suitable Python 2.7 installer from the following website and follow the instructions of the installation program to install Python 2.7 on your system.

https://www.python.org/downloads/

**NOTE**
The biosignalsplux Python API is designed for Python 2.6 and 2.7. Installing Python 3.X will not work with the API.

## 2.2 MATLAB Requirements

**NOTE**
The biosignalsplux Python API is designed for Python 2.6 and 2.7. Installing Python 3.X will not work with the API.

Python use is supported in MATLAB versions starting with the **2014b** release and newer releases.

If a supported release has not yet been installed on your computer, login into your MathWorks account and download and install a suitable MATLAB version on your system.

To check if MATLAB recognizes the installed Python version, open MATLAB and enter the `pyversion` command into the command window which will return information about the Python interpreter being used by MATLAB.

```
>>  pyversion
                    version:  '2.7'
                 executable:  'C:\Python27\python.exe'
                    library:  'C:\windows\systems32\python27.dll'
                       home:  'C:\Python27'
                   isloaded:  0
```

Please follow the information in the troubleshooting section of this guide, if this command does not return any information (parameter values are empty) or if the used Python interpreter is not compatible with the biosignalsplux API (version 2.6 or 2.7).

## 2.3 biosignalsplux API

The PLUX Python API brings to Python applications all the functionality of PLUX devices and can be downloaded here:

http://biosignalsplux.com/downloads/api/PLUX_API_Python2.7.zip

It is recommended to review the official API documentation for detailed information about the available classes, methods, and functions to take advantage of all the available biosignalsplux features:

http://biosignalsplux.com/downloads/api/PLUX_API_Python2.7/

After downloading the API, unzip the file and copy the **plux.pyd** file from the appropriate platform folder (win32 or win64) into one of the available Python search paths. (Recommendation: place the **plux.pyd** inside the **.\lib\site-packages** directory of your Python installation path (e.g. C:\Python27\lib\site-packages).

# 3 Using the Python API in MATLAB

The following instructions were designed to show how to connect, set up, and disconnect your biosignalsplux device as well as how to start and stop signal acquisitions with the Python API.

**NOTE**
This process is done by following the general syntax for using Python code in MATLAB, using the built-in py.* command:

```
py.module.function()
```

**NOTE**
It is not required to import the PLUX API using the Python `import` statement as MATLAB will automatically load the API whenever its content is used in the MATLAB code. Using this statement can cause issues with your code, as the MATLAB `import` statement does not share the same behaviour as the Python statement.

For more information, please read the official MATLAB article on this topic:

https://de.mathworks.com/help/matlab/matlab_external/python-import-and-matlab-import-commands.html

## 3.1 Setting Up Help-Script (Override Callbacks)

In order to be able to use the API, some custom classes may need to be created which inherit from the API device classes to customize and overwrite callback functions.

For example, the `onRawFrame` callback is called by message loop when a real-time data acquisition frame is received from the device. In order to receive data frames, an application must derive `plux.SignalsDev` class (or any of its derived classes) to a new class and override this method in the new class. In order to set a maximum acquisition time or to manage the frames during acquisitions.

An example of such custom classes can be found in the Python scripts that come with the Python API. It provides two custom classes, `PluxMemory` which inherits from the `plux.MemoryDev` class for biosignalsplux devices with internal memory and

`PluxAcquisition` which inherits from the `plux.SignalsDev` class (more information later). Both classes have new `onRawFrame` methods which limit the signal acquisition to 30.000 Frames (30 seconds at 1000Hz sampling rate) and overwrite the original method(s) of the parent classes.

Customization of class methods is required in order for you to take the most advantage of the PLUX API.

**NOTE**
The upcoming examples are based on custom classes stored in a *biosignalsplux.py* named file as shown in the API examples.

## 3.2 Finding & Connecting Devices

The PLUX API connects to biosignalsplux devices using the MAC-address of the device. If known, the MAC-address of your device can be used directly to connect to your device (see the back of your device). Alternatively, the `plux.BaseDev.findDevices()` function can be used to search for available biosignalsplux devices and returns a Python tuple with the addresses of the found devices.

Using this function in the command window will provide the following output:

```
>> py.plux.BaseDev.findDevices( )

ans =
       Python tuple with no properties.

                  (('BTH00:07:80:D8:AA:B7', 'biosignalsplux'), )
```

When detected and selected, the MAC-address is used to create a Python object of the available PLUX API device classes.

For this, it is recommended to use one of the available device classes depending on the device you are using to take advantage of all the features:

- `plux.SignalsDev` is designed for devices without internal memory
- `plux.MemoryDev` is designed for devices with internal memory

When using a device without internal memory, the MATLAB code should now be similar to the code below.

```
% Connect to the device without internal memory
% (inherited from plux.SignalDev( ))
dev = py.biosignalsplux.PluxAcquisition('00:07:80:D8:AA:A7')
```

When using a device with internal memory, the MATLAB code should be similar to the code below.

```
% Connect to the device with internal memory
% (inherited from plux.MemoryDev( ))
dev = py.biosignalsplux.PluxMemory('00:07:80:D8:AA:A7')
```

If no error occurs, the command window should display an output similar to the following (if the output is not suppressed):

```
dev =
       Python tuple with no properties.

                  (('BTH00:07:80:D8:AA:B7', 'biosignalsplux), )
```

### 3.3 Setting Up Devices

After successfully connecting to your device, the device can be set up for signal acquisition. This is done using the `plux.SignalsDev.start` or `plux.MemoryDev.start` methods (or in this case `biosignalsplux.PluxAcquisition.start` or `biosignalsplux.PluxMemory.start`) which configure the device and start the signal acquisition.

For this, it is recommended to define variables with values for the sampling rate, the selected channels, and the sampling resolution. Note, that this function will also start the signal acquisition.

The **sampling rate** supports integer values for the frequencies 100, 200, 300, 400, ..., 1000Hz[1].

[1] biosignalsplux devices support other frequencies with newer firmware, however, these frequencies are not

The **channels** are configured by an integer representing an 8-bit bitmask of the device ports to acquire. The least significant bit corresponds to port 1, the next bit to port 2, etc. (example: activating channel 1 and 3 only: 0b00000101). Bitmasks can be used by converting them into the equivalent integer values with the `bin2dec()` function.

The **sampling resolution** is an integer value of 8 or 16 which sets 8-bit or 16-bit resolutions respectively.

**NOTE**
MATLAB stores numeric values as double type by default, which can cause confusion when trying to pass integer values to Python functions. The use of the `int8( )`, `int16( )`, `int32( )` and `int64( )` functions are required to prevent such issues with the PLUX API.

An example configuration code can be seen below:

```
% 1000Hz sampling rate
sampling_rate = int16(1000);

% 16-bit sampling resolution
sampling_resolution = int8(16);

% Acquisition channels (here: 2 & 4)
acq_channels = int8(bin2dec('00001010');

% Start acquisition
dev.start(sampling_rate, acq_channels, sampling_resolution)
```

If followed correctly, your MATLAB scripct should now be able to communicate and to control your biosignalsplux device and you should now be ready to work with the acquired signal data in MATLAB.

compatible with current API versions.

# 4 Troubleshooting

## 4.1 MATLAB can't find my device(s)

Available PLUX devices can be found using the `plux.BaseDev.findDevices()` function, however, in some cases no devices can be found although the devices are turned on and the Bluetooth module of your computer is turned on as well.

To prevent this issue, it is recommended to pair your device(s) with your Windows operating system first before trying to use them in MATLAB.

Follow the instructions provided in the biosignalplux user manual to learn how to correctly pair biosignalsplux devices with your computer:

http://biosignalsplux.com/downloads/biosignalsplux_User_Manual_v.1.0.pdf

## 4.2 MATLAB can't find the Python interpreter or is using a wrong Python version

Using the `pyversion` command can return empty parameter values if a wrong Python version of the Python interpreter is installed on your machine or if you have installed Python in an uncommon location. In this case, the `pyversion` command will return the following information:

```
>>  pyversion
                    version:  ''
                 executable:  ''
                    library:  ''
                       home:  ''
                  isloaded:  0
```

Confirm that a 32-bit Python 2.7 interpreter is installed on your machine when using a 32-bit MATLAB release or that a 64-bit Python 2.7 interpreter is installed when using a 64-bit MATLAB release.

MATLAB might not be able to find Python as it might be installed in an uncommon location. To manually set the correct location of the Python interpreter enter the following command into your command window, with <path> being substituted by the correct path to your Python Interpreter.

```
>>  pyversion <path>
```

MATLAB should now be able to find the correct Python paths.

```
>>  pyversion
                    version:  '2.7'
                 executable:  'C:\Python27\python.exe'
                    library:  'C:\windows\systems32\python27.dll'
                       home:  'C:\Python27'
                   isloaded:  0
```

### 4.3 MATLAB does not recognize the py. Command

(see information provided in the previous answer)

### 4.4 MATLAB Error – Undefined variable 'py' or function 'py.command'

Errors of different natures can occur when trying to call python code and will trigger this error message. In this case, it is recommended to check whether any of the previous solutions can solve the issue or if it can be solved with some of the solutions provided by MathWorks:

https://de.mathworks.com/help/matlab/matlab_external/undefined-variable-py-or-function-py-command.html

Additionally, it is recommended to check if the required module is accessible outside MATLAB. This can be done by importing and testing the module in the Python IDLE.

### 4.5 Python Error: TypeError: integer argument

MATLAB stores numeric values as double type by default, which can cause confusion when trying to pass integer values to Python functions. The use of the `int8()`, `int16()`, `int32()` and `int64()` functions are required to prevent such issues with the PLUX API.

More information on datatype compatibilities can be found in the official MATLAB article:

https://de.mathworks.com/help/matlab/python-data-types.html

### 4.6 Python Error: Missing DLL files when loading PLUX API

This issue may occur when using older versions of the PLUX API (2015 and older) which require additional external libraries which cannot be found by Python or MATLAB. It is recommended to use the newest release of the PLUX API to prevent such issues.

Download the newest version from the biosignalsplux API website or contact our support team ( support@plux.info ) to get access to the most recent version if you continue having the same issue.

http://biosignalsplux.com/en/software/apis

## 4.7 Python Error: RuntimeError: The communication port does not exist or it is already being used.

This error can occur if the communication between your computer and your biosignalsplux has not been closed and a new attempt to connect to your device is being made.

This can happen, for example, if a script crashes during its runtime without reaching the `dev.close()` method to disconnect from the biosignalsplux device and a new connection attempt follows when rerunning the script.

If the workplace variables are still available, the device can be disconnected by entering the `dev.close()` into the command window. Alternatively, turn your device off and on again and restart your script.