

# **BioLink**

A tool for synchronized psycho-physiological  
and behavioural data acquisition.

Software Manual

Julian Schneider & Marcus Cheetham

## **Preface**

This document is an instruction manual for BioLink. This software is intended for use with two well-known stimulus presentation packages (Presentation and PsychoPy) and a set of PLUX research kits (i.e. devices and software) for physiological data acquisition, analysis and visualization for a large range of different physiological signals. These kits are available with a range of open source software that are intended to help the researcher tailor data collection, analysis and visualization for own purposes. Data synchronization tools are also available with the research kits. To date, these tools do not support the mapping (i.e. synchronization and labelling) of physiological responses to particular stimulus events in the observational data. BioLink overcomes this by enabling the mapping and storage of raw physiological and observational data in one file. The mapped data may be visualized with BioLink and extracted for further processing and analysis with other tools. This manual explains how to access, set up and use BioLink.

December, 2017

Julian Schneider and Marcus Cheetham

Department of Internal Medicine, University Hospital Zurich, Zurich, Switzerland

Unit of Neuropsychology, Department of Psychology, University of Zurich, Switzerland

# Contents

1 Introduction.....	3
1.1 Getting Started.....	4
2 User Interface.....	5
2.1 Settings.....	5
2.2 Experiment.....	6
2.3 Other Functions.....	7
2.4 Viewing Previously Recorded Data.....	7
2.5 Logfile Format.....	7
3 Serial Events.....	8
4 Presentation® Setup for Serial Events.....	9
4.1 Hardware Setup.....	9
4.2 Port Setup in Presentation®.....	9
4.3 Presentation® Experiment Procedure.....	10
5 Integrating with PsychoPy experiments.....	11
6 Setup for Development.....	11
6.1 Development Notes.....	12
6.2 Setting up your own Python Installation.....	12
7 Developing New Extensions.....	13
7.1 How to integrate PsychoPy code.....	13
8 References.....	14

## 1 Introduction

BioLink enables the integrated acquisition, synchronization and labelling of stimulus events acquired from observational data, using the stimulus presentation packages Presentation® [1] or PsychoPy [2], with physiological data acquired using a biosignalsplux [3] research kit. The biosignalsplux kits include devices and software. The devices are referred to as the “PLUX device” throughout this manual. BioLink bridges a specific gap in the present range of biosignalsplux synchronization tools. Harmonized marking and labelling of stimulus events and physiological data is accomplished by synchronizing the timestamps of stimulus events with the physiological data in one timeline. The raw data is stored unprocessed and can be analysed using any tool that supports the import of .txt files.

BioLink is a pure Python application, providing an easy to use graphical user interface (GTK3 based). It is distributed under LGPLv3 license [4]. For compatibility with PsychoPy, it is built using Python 2.7 32-Bit edition.

BioLink has been tested on Windows® 7 and 10. It could be readily be adapted to run on other platforms, provided that the data acquisition hardware (PLUX) supports these.

## 1.1 Getting Started

The following steps show how to set up BioLink on a Windows® platform, using the binary release of BioLink downloadable from github.

- Download and install OpenSignals for **Windows 32-bit** from the biosignalsplux homepage:  
<http://biosignalsplux.com/en/software>  
*Note:* You need the 32-bit installation irrespective of whether you are operating a 32-bit or a 64-bit version of Windows®. This is because BioLink runs on a 32-Bit Python installation. In order for BioLink to be able to access the hardware driver integrated in OpenSignals, these versions need to match.  
Please ensure that OpenSignals is installed in its standard installation path ('C:\Plux').
- Download the binary release zip-file from <https://github.com/BioLinkJ/BioLink/releases>
- Unpack the zip file in the desired installation directory (recommendation: directly in C:\ or in your user directory).
- Pair PLUX device (before using it for the first time, also see OpenSignals Manual [5])
  - make sure the Bluetooth interface is turned on as well as the PLUX device.
  - click on the Bluetooth symbol in the task list (bottom right of screen) and choose “Add a Bluetooth device”.
  - select / add the device called “biosignalsplux” (PIN: 123).
- BioLink is started by launching (double-click) the file ‘start\_BioLink.bat’ in the unpacked BioLink directory.

Launching BioLink for the very first time can take up to a few minutes, as all the Python source code will be compiled at first run.

*Note:* At first run, the message “Error loading '../settings.json': [Errno 2] No such file or directory: '../settings.json'” will be shown in the message area. This is normal and will only occur the very first time BioLink is run, as the settings are initialised with default values and saved to the ‘settings.json’ file.

See Section 2 for an explanation of the user interface and how to start an experiment.

## 2 User Interface

The functionality of the graphical user interface is explained in this Section.

### 2.1 Settings

How to configure the settings (see Fig. 1):

- **Serial Port:** Address of the serial port (to receive serial events or subject id). Default: COM1  
→ to find available serial ports click on ‘find ports’. The ports found will be listed in the message area at the bottom of the window. Do not select serial port numbers that are stated as Bluetooth devices, as they are used for the PLUX device.
- **Plux MAC:** Address of PLUX device  
→ to find available PLUX devices click on ‘find devices’. The devices found will be listed in the message area at the bottom of the window.  
*For test purposes:* enter ‘dummy’. No acquisition device will be opened and only dummy data in the shape of sinewaves will be generated.
- **Channel Names:** This field specifies what sensors are attached to which channel of the PLUX device (and which channels are therefore activated). The format is *channel number : name*. The different channels are separated by commas. It is possible to have non-consecutive channel numbering (e.g. 1:ECG, 2:EDA, 4:BVP → in this case, channels 1,2,4 will be used).
- **Sample rate:** The sample rate for the PLUX device can be: 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000 (samples pre second).
- **Max Duration:** This is the maximum duration the experiment can take. This needs to be specified in advance so that BioLink can allocate enough memory for the whole experiment at the beginning. Max duration can be 1 to 480 minutes. The experiment will automatically terminate after the max duration has elapsed.  
*Note:* The higher the Max Duration setting, the more memory is preallocated at experiment start and the more working memory (RAM) will be needed by BioLink.
- **Life Plot (see Fig. 2):** This can be disabled to save processor power (‘Paint Life Plot’). Automatic reopening of the life plot window during the experiment can be disabled (‘Reopen Life Plot if closed’).
- **Extensions:** Registered extensions can be selected (see Section 7).

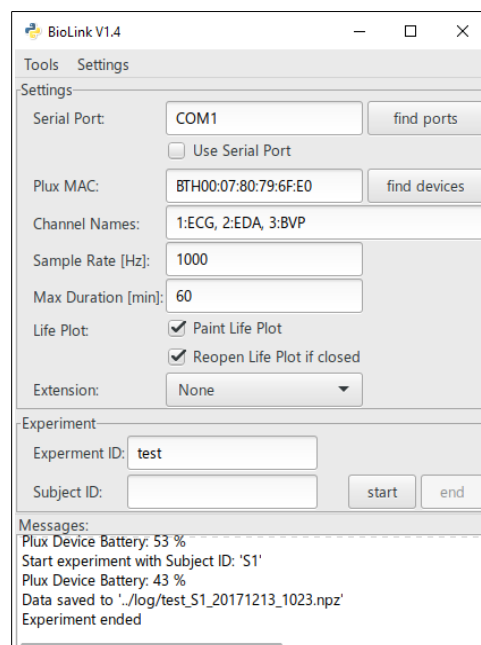


Fig. 1: BioLink Control Window

## 2.2 Experiment

Requirements to start an experiment:

- Experiment ID: This can be any name identifying the experiment.
- Subject ID: To start the experiment, a subject ID is needed. If this field is left empty and a connection to the serial port (specified by Serial Port setting) is established, the subject ID can also be received from another computer connected via the serial port (see Section 3).

*For test purposes:* Enter 'nolog' in order to generate no log files.

To start the experiment click on 'start'.

- If no Subject ID is entered and the serial port connection is established, BioLink waits for entry of the subject ID via the serial port (see Section 3) after 'start' is clicked.
- Once a subject ID is received and the experiment starts, the Life Plot opens (see Fig. 2, only if activated), showing the current data received from the acquisition device. The different plots can be zoomed independently of one another by clicking on the 'zoom rectangle' button (toolbar at bottom of window).

Experiment is ended by one of three events:

- Click on button 'end'.
- End event is received via serial port (see Section 3).
- The max duration time is over.

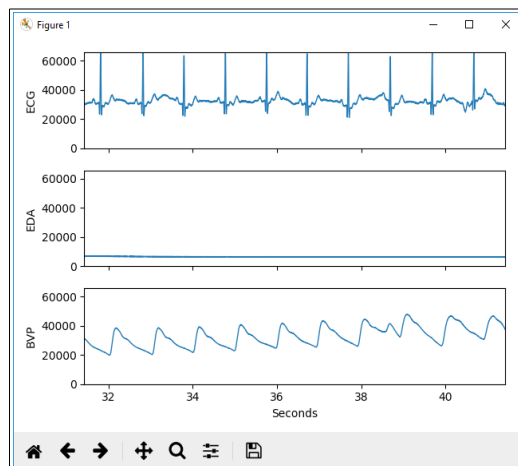




Fig. 2: BioLink Life Plot Window

## 2.3 Other Functions

- Menu Tools > Convert log to txt: This converts an existing log file into text format (tab-separated columns) for analysis using other tools.
- Menu Settings > Default Values: This resets all settings to their default value

## 2.4 Viewing Previously Recorded Data

The recorded data can be viewed as follows (see Fig. 3):

- Menu Tools > Plot BioLink Data
- Choose the logfile that you want to view (filename ending in .npz)
- You can zoom in and out of the plot with the
  - pan/zoom tool 
    - left click and drag to scroll
    - right click and drag to zoom in x or y direction on any graph to zoom in or out in x or y direction
  - or with the zoom rectangle tool 
    - zoom to a rectangle in any graph

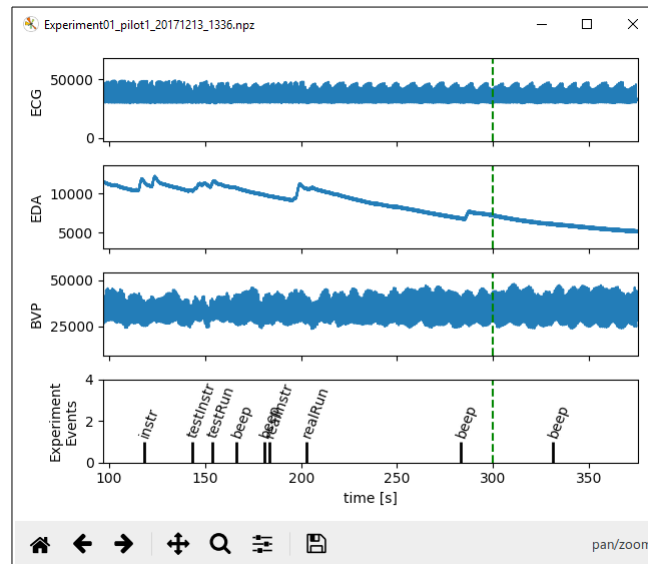


Fig. 3: BioLink Displaying Previously Recorded Data

## 2.5 Logfile Format

For every experiment, two files are saved in the log directory. One file has a name with the ending ‘.npz’. This file contains all the data in numpy compressed format. It can be loaded with the `numpy.load` function in Python (NumPy). It contains three arrays: `channelHeader`, `bioData`, `eventData`.

The other file has a name with the ending ‘.json’. It contains all experiment information such as subject ID or the acquisition data and time. The json file can be viewed with any text editor.

For data analysis in other tools, logfiles can be exported in text format with tab-separated columns (see Section 2.3).

### 3 Serial Events

Events from an experiment procedure running on another computer can be sent to BioLink using a serial interface (RS232). See Section 4 for an introduction how to set it up with Presentation® running on another computer.

All serial events have to be terminated by sending a newline character ('\n'). The length of the event strings should not be longer than 16 characters (= letters). Therefore, it is recommended to use abbreviations.

The events are always logged along with the current time of the acquisition device.

Possible commands are:

'#ID:subject_id\n'	Optional: transmit <i>subject_id</i> before experiment starts.
'event\n'	Any event string, not longer than 16 letters (plus '\n' termination)
'#END\n'	End experiment

Serial port configuration:

- Baudrate: 115200
- Parity: none
- Data Bits: 8
- Stop Bits: 1



## 4 Presentation® Setup for Serial Events

Presentation® [1] experiments can be linked with BioLink. Events in the experiment procedure, such as the presentation of a new stimulus, are forwarded to BioLink in order to place a mark and timestamp in the BioLink data record; the mark is also labelled with the name of the event that is coded in Presentation®. This synchronises the Presentation® experiment with the timeline of the physiological data, allowing for precise, automated analysis. The subject ID entered in presentation is also sent to BioLink so that it does not have to be entered in both programs individually (thus eliminating a potential source of error). At the end of the experiment procedure, an end tag can be sent so that BioLink automatically stops acquisition when the experiment procedure is complete.

### 4.1 Hardware Setup

The PLUX device and BioLink usually run on a different computer than Presentation®, as it is not advised to have other processes running in parallel with Presentation® on the same machine. A serial interface (RS232) is used for communication between the computers as this is easy to set up and comes with relatively low and predictable latency delays. If one of your computers (or both) do not provide a serial interface, a USB to RS232 adapter can be used. To connect the computers, a Null-modem-cable is needed (see Wikipedia [6]).

### 4.2 Port Setup in Presentation®

To link your Presentation® [1] experiment with BioLink, the Presentation® output port settings have to be set up as follows:

- In Presentation®, under Tab “Settings”, click on “Port” and add an “Output Port”. Remember the output port number assigned by Presentation® for later.
- Select a port name starting with “COM” (e.g. “COM1”), as it is a serial port.
- Click on “Properties” next to the selected port
  - The configuration should be:
    - Type: Serial Port
    - Rate: 115200
    - Parity: none
    - Data Bits: 8
    - Stop Bits: 1
    - other settings: off / disable

### 4.3 Presentation® Experiment Procedure

Insert the following initialisation code at the beginning of the pcl section of the Presentation® experiment (below line `begin_pcl`; in the .sce file). Typically, other initialisation steps are done at the top of the pcl section, too.

At first, the serial port is opened. The argument given to function `output_port_manager.get_port` is the output port number the port was set up as in Presentation® port setup.

The functions `biolink_event`, `biolink_send_subject_id` and `biolink_end` are defined here and will be used throughout the experiment procedure.

```
#BioLink initialisation
#serial port is output port 1 in Port settings, adapt if settings change
output_port serial = output_port_manager.get_port(1);

#function to send event to BioLink.
#The sort_msg identifying the event should not be longer than 16 char (letters)
sub biolink_event(string short_msg)
begin
    serial.send_string( short_msg + "\n");
end;

#function to send subject ID to BioLink
#BioLink then uses the subject ID that has been entered in Presentation
#Experiment must be started in BioLink without specifying a subject ID.
#BioLink then waits to receive the subject ID on the serial port.
#BioLink starts data acquisition after receiving the subject ID.
sub biolink_send_subject_id
begin
    string subjectName = logfile.subject();
    if subjectName == "" then
        subjectName = "NA";
    end;
    serial.send_string("#ID:" + subjectName + "\n");
end;

#function to send end tag to BioLink to stop acquisition of physiological data
sub biolink_end
begin
    serial.send_string("#END\n");
end;
```

The subject ID from Presentation® may be forwarded to BioLink by inserting this function call at the beginning of your procedure code (in the pcl section, after all initialisation code):

```
biolink_send_subject_id();
```

At any time point in the experiment procedure, a timestamp may be produced in the BioLink data record by inserting this function call. Replace “myevent1” with your event text that uniquely identifies the point in your procedure. Note that the event text cannot be longer than 16 chars (letters).

```
biolink_event("myevent1");
```

It can be very useful to generate an event text that includes an integer variable from presentation:

```
biolink_event("myevent" + string(integer_variable));
```

At the end of your procedure, call `biolink_end` as a last command. This sends an end tag to BioLink so the data acquisition is stopped automatically.

```
biolink_end();
```

## 5 Integrating with PsychoPy experiments

PsychoPy [2] experiments can be integrated into BioLink extensions. In order to develop an extension that uses PsychoPy, install PsychoPy into the same Python installation (see Section 6.2). Follow the steps for manual installation on the PsychoPy homepage:

<http://psychopy.org/installation.html#manual-install>

See Section 7 for instructions on how to incorporate your PsychoPy experiment code into a new BioLink extension.

Alternatively, it is possible to run a stand-alone PsychoPy experiment on the same computer (independent program) or a different computer and link it using the serial interface. Please refer to Section 3 for instructions on how to format the messages sent through the serial interface and to Section 4.1 for a description of the hardware setup. The described setup can also be used to link with the same computer. In that case, PsychoPy and BioLink will be using different port numbers (e.g. “COM1” and “COM2”) for the two serial interfaces of the same computer that are connected using the Null-modem-cable.

Please refer to the PsychoPy reference manual [7] and the pySerial documentation [8] for examples on how to use the serial port from Python.

## 6 Setup for Development

BioLink was developed using a WinPython 2.7 32-bit installation.

To develop extensions or improve BioLink, the following are required:

- Install Python 2.7 or WinPython 2.7 32-bit (see Section 6.2).
- Setup the Python development environment (IDE) of your taste.  
(We are using Eclipse with PyDev plugin.)
- Download of source code from github at: <https://github.com/BioLinkJ/BioLink>  
or clone git repository from <https://github.com/BioLinkJ/BioLink.git>  
into your workspace / project directory.
- Import project into your IDE.

## 6.1 Development Notes

This section contains a few notes on certain choices and thoughts made during development that might be useful to know for future developers.

- BioLink is based on a Python 2.7 32-bit version. The reason for this is that 32 bit Python 2.7 is compatible with PsychoPy and biosignalsplux.
- The settings of configured in the user interface are stored in the file 'settings.json' in the parent directory of the src directory (in JSON format) and are loaded upon program start. That file is created and initialised with default values if it does not exist.  
The file path is determined by variable `SETTINGS_FILE_PATH` in 'Controller.py'.
- The user interface is GTK3 based and was designed using the tool Glade. The design data determining the look of the main window is loaded from file 'BioLink\_wnd.glade'.
- Plots, data management and data storage are based on the widely applied Python libraries matplotlib [9] and numpy [10].
- PyQt is a more efficient matplotlib backend for the Life Plot but is not included in the binary release (zip-file) to keep the filesize to a minimum. See Section 6.2 for instructions on how to change the backend used by matplotlib.
- Every source file contains a brief description for its functionality below the license header.
- Memory for the data to be recorded is preallocated in RAM at experiment start. This is done to prevent unforeseeable delays during data recording.  
Data is stored to the hard drive when the experiment ends.

## 6.2 Setting up your own Python Installation.

Use either a Python 2.7 32-bit or WinPython 2.7 32-bit. To install the required additional packages, type the following command in the command prompt (launch 'WinPython Command Prompt.exe' in your WinPython install directory):

```
pip install numpy scipy matplotlib pypiwin32 pyserial future
```

As the user interface is based on GTK3, PyGObject is needed.

The Windows-version can be downloaded at: <https://sourceforge.net/projects/pygobjectwin32/>

In case WinPython is being used, PyGObject needs to be installed for a portable installation and the path of the Python directory needs to be selected (e.g. 'WinPython\python-2.7.13').

It is recommended that only the components 'Base packages' (in the very first list of components) and 'Glade' (in the third list of components) are installed. Glade was used to design the BioLink user interface.

Now change the default matplotlib-backend to GTK3:

- in your Python directory, open file  
'python-2.7.xx\Lib\site-packages\matplotlib\mpl-data\matplotlibrc' in a text editor
- find line  
    backend        : Agg  
and change it to  
    backend        : GTK3Agg

If you are going to develop an extension that uses PsychoPy, you should install PsychoPy into the same Python installation. Follow the steps for manual installation on the PsychoPy homepage:

<http://psychopy.org/installation.html#manual-install>

## 7 Developing New Extensions

An interface for future extensions is provided. The class `ExtensionInterfaceBackend` provides an interface to feed Events to the BioLink log file and also to get the current physiological data for further processing or interaction. The extension base class `ExtensionBase` provides all base functionalities for all extensions like writing an extension log or initializing a data processing thread (if needed) that is fed with the data stream from the data acquisition device. In addition to the events from the extension being recorded alongside with the physiological data (from PLUX device), extensions typically create a separate log-file with only extension related data.

New extensions:

- The python file should be placed in the `src/Extensions` directory.
- The extension class should be registered in the file `src/Extensions/__init__.py`:  
import:  

```
from pythonfile.py import ExtensionClass
```

  
register all extensions in variable `extensionClasses`:  

```
extensionClasses = [ExtensionTemplate, ExtensionClass ]
```
- The extension class should inherit from `ExtensionBase` and use functions of `self.eib` (instance of `ExtensionInterfaceBackend` class) to interface with BioLink.
- See the template extension (with file ‘`template.py`’) for an example.
- Please read all the comments in files ‘`ExtensionBase.py`’, ‘`template.py`’ as well as the comments for the class `ExtensionInterfaceBackend` in ‘`ExtensionInterface.py`’ as these give a good introduction to the extension framework.

A minimal extension class should:

- Inherit from class `ExtensionBase`.
- Initialise class variables `logColumnHeader` with a list of column names for the extension log-file and `extensionName` with the extension name the user will see.
- Implement (override) all the methods commented with ‘To be overridden by extension.’ in `ExtensionBase.py`, namely: `extMainLoop` and `onExtEndRequest` (`onBioDataFrame` is only necessary if the new extension will receive data from the acquisition device in real time)
- Use the methods
  - `self.eib.putEvent` to send events with a comment string (timestamp is taken automatically) to BioLink
  - `self.logAppendLine` to append a line to the log-file of the extension.
  - `self.startBioDataProcessing` to start receiving data from the acquisition device in real time (if needed, `onBioDataFrame` method must be implemented)
  - `self.eib.endExperiment` to indicate to BioLink that the experiment is done. BioLink will then end the experiment properly which results in a call to the method `onExtEndRequest`.

### 7.1 How to integrate PsychoPy code

The core of every extension is the method `extMainLoop`. This method is run within a separate sub-process and should not end (i.e. should block) until the experiment is ended by BioLink with a call to `onExtEndRequest`.

The code for the main execution thread for any PsychoPy (or other) experiment can be inserted in this method. The PsychoPy procedure code can simply be inserted as the code for method `extMainLoop` (along with the needed import statements in the header of the extension source file).

## 8 References

- [1] “Neurobehavioral Systems.” [Online]. Available: <https://www.neurobs.com/>. [Accessed: 13-Dec-2017].
- [2] J. W. Peirce, “Generating stimuli for neuroscience using PsychoPy,” *Front. Neuroinformatics*, vol. 2, 2009.
- [3] “biosignalsplux | wearable body sensing platform - Research Kits.” [Online]. Available: <http://biosignalsplux.com/en/products/lab-kits>. [Accessed: 13-Dec-2017].
- [4] “GNU Lesser General Public License v3.0 - GNU Project - Free Software Foundation.” [Online]. Available: <https://www.gnu.org/licenses/lgpl-3.0.en.html>. [Accessed: 20-Dec-2017].
- [5] “opensignals (r)evolution user manual.” [Online]. Available: [http://www.biosignalsplux.com/downloads/OpenSignals\\_\(r\)evolution\\_User\\_Manual-print.pdf](http://www.biosignalsplux.com/downloads/OpenSignals_(r)evolution_User_Manual-print.pdf). [Accessed: 13-Dec-2017].
- [6] “Null modem - Wikipedia.” [Online]. Available: [https://en.wikipedia.org/wiki/Null\\_modem](https://en.wikipedia.org/wiki/Null_modem). [Accessed: 14-Dec-2017].
- [7] “psychopy.serial - functions for interacting with the serial port — PsychoPy v1.8.” [Online]. Available: <http://www.psychopy.org/api/serial.html>. [Accessed: 21-Dec-2017].
- [8] “Welcome to pySerial’s documentation — pySerial 3.0 documentation.” [Online]. Available: <http://pythonhosted.org/pyserial/>. [Accessed: 21-Dec-2017].
- [9] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” *Comput. Sci. Eng.*, vol. 9, no. 3, pp. 90–95, 2007.
- [10] “NumPy — NumPy.” [Online]. Available: <http://www.numpy.org/>. [Accessed: 20-Dec-2017].