Enhancing Biosignal Data Analysis with LSL and BioLab

Ján Stehlík, Michal Kubaščík, Peter Šarafín

***Abstract*—** Efficient collection and analysis of biological signals are essential for progress in neuroscience, physiology, and related fields. These capabilities address the increasing complexity of modern experimental setups, allowing researchers to focus on insights rather than technical challenges. The Lab Streaming Layer provides a comprehensive system for managing time-series data, offering solutions for networking, precise time synchronization, and real-time access across devices such as electroencephalography and eye-tracking tools. Complementing this, the Biosignals Laboratory simplifies the process of streaming and analyzing signals online through configurable scripts. Together, these tools streamline workflows, enabling seamless real-time insights and fostering innovation in the study of biological signals. This paper examines their capabilities, practical applications, and future potential.

***Keywords*—** Biological signals, electroencephalography, data synchronization, signal processing.

I. INTRODUCTION

In modern research, the ability to seamlessly collect, synchronize, and analyze biosignal data is critical for advancing our understanding of human physiology and behavior. The Lab Streaming Layer (LSL) provides a robust framework for unified data collection in experimental settings, enabling real-time access, time synchronization, and centralized recording[1]. This open-source system supports diverse devices, from EEG and fNIRS to eye-tracking equipment, and integrates seamlessly across multiple programming languages, making it a versatile tool for researchers across disciplines.

Complementing LSL is the Biosignals Laboratory (BioLab), a project designed to simplify signal streaming and data analysis. By leveraging configurable scripts, Biolab streamlines the creation of data streams, providing users with accessible tools to visualize, analyze, and process incoming data in real-time. The integration of BioLab with LSL offers unparalleled flexibility, allowing researchers to connect to and manage biosignal streams effortlessly.

This article delves into the key features of LSL and Biolab, provides practical examples of their applications, and explores potential future developments that could further transform biosignal research. Whether you are an academic, a developer, or a professional in the field, these tools present exciting opportunities to enhance data-driven discoveries and innovation.

II. LAB STREAMING LAYER

The Lab Streaming Layer (LSL) is a comprehensive framework designed to unify the collection and management of measurement time series in research experiments. Its robust system facilitates the seamless integration of various data sources, addressing critical aspects such as networking, precise time synchronization, near-real-time data access, and optional centralized collection. By resolving common data synchronization and integration challenges, LSL significantly improves the efficiency of experimental workflows, particularly in multi-device environments. This centralized capability allows for efficient viewing, recording, and archiving of data directly onto a disk, making LSL an essential tool in research workflows[2].

Ján Stehlík, University of Zilina, Zilina, Slovakia (e-mail: stehlik16@stud.uniza.sk)
Michal Kubaščík, University of Zilina, Zilina, Slovakia (e-mail: michal.kubascik@fri.uniza.sk)
Peter Šarafín, University of Zilina, Zilina, Slovakia (e-mail: peter.sarafin@fri.uniza.sk)

This versatility makes LSL invaluable for a range of applications, from monitoring neural activity to studying eye movements, enhancing both research accuracy and efficiency. To visualize its functionality, imagine a network with two primary applications: an outlet and an inlet. The outlet serves as the source of streaming data, broadcasting its availability to the network. The inlet, acting as the receiver, identifies and connects to the outlet. Once connected, the outlet buffers data and transmits it upon the inlet's request. This architecture ensures a smooth flow of data between connected devices, even in demanding research scenarios.

What makes LSL particularly versatile is its wide compatibility with numerous programming languages and a diverse range of supported devices. These include cutting-edge tools like EEG systems, functional near-infrared spectroscopy (fNIRS), and eye-tracking devices. This broad support allows researchers across multiple disciplines to adopt LSL in their experimental setups, simplifying the challenges of integrating heterogeneous data sources into a cohesive system[3][4].

By leveraging LSL, researchers can focus on data-driven insights rather than the technical intricacies of data collection and synchronization, paving the way for innovative breakthroughs in neuroscience, physiology, and other fields [5].

III. BioLab

The Biosignals Laboratory (Biolab) is an innovative project designed to facilitate the streaming of signals and data for real-time online analysis. By employing configuration files and scripts, Biolab enables users to efficiently create and manage stream outlets, simplifying the process of collecting and analyzing biosignal data. This structured approach allows researchers to focus on data interpretation and application rather than dealing with complex setup procedures.

A. Examples

Biolab examples showcase practical implementations of LSL and Biolab, focusing on how to establish inlets and efficiently extract and process received data streams. There are currently 3 example programs, which are stream search, select stream and plot data. Each example demonstrates a specific aspect of these tools, making it easier for users to understand and apply their functionalities in real-world scenarios.

1) Stream search

This program allows the user to search for available streams. Upon running the program, the user is asked to type into the terminal the type of stream that they wish to search for. If left empty, the program will search for all available streams, otherwise it will search for streams of that type.

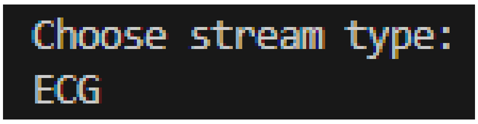

Fig. 1 Example of choosing stream type

If any number of streams are found, their name, type, channel count and user id (UID) will be written on the terminal. If none are found, the message “No streams were found” will be printed on the terminal.

```
Looking for EEG streams...
2024-11-24 15:53:22.435 (   4.539s) [         5F5902C]      netinterfaces.cpp:36    INFO| netif '{A81891A8-4A91-4F
07-8D4B-90AFE7B4678A}' (status: 1, multicast: 1
2024-11-24 15:53:22.435 (   4.539s) [         5F5902C]      netinterfaces.cpp:36    INFO| netif '{3285BD52-5F4F-4F
65-AB66-3B82E8A78F64}' (status: 2, multicast: 1
2024-11-24 15:53:22.435 (   4.540s) [         5F5902C]      netinterfaces.cpp:36    INFO| netif '{8CA27D4A-24D7-40
48-9D51-9D6F5774539E}' (status: 2, multicast: 1
19-8342-F3E4A342BC4B}' (status: 2, multicast: 1
2024-11-24 15:53:22.436 (   4.540s) [         5F5902C]      netinterfaces.cpp:36    INFO| netif '{85C9A568-0283-49
42-B890-F717008A29DF}' (status: 2, multicast: 1
2024-11-24 15:53:22.436 (   4.540s) [         5F5902C]      netinterfaces.cpp:36    INFO| netif '{1AA6237C-273B-11
EC-B1C8-806E6F6E6963}' (status: 1, multicast: 1
2024-11-24 15:53:22.436 (   4.540s) [         5F5902C]      netinterfaces.cpp:58    INFO|         IPv6 ifindex 1
2024-11-24 15:53:22.436 (   4.540s) [         5F5902C]         api_config.cpp:270   INFO| Loaded default config
1 were found:
BioSemi   EEG   8   f27926a2-6814-40e9-96be-3af52b6cfc63
```

Fig. 2 Console output of found stream

2) Select stream

This program allows the user to connect to a stream by their name, user id and type. At first, the program asks the user to type the name, UID and type into the terminal. The program then attempts to find the streams based on the user's input.

```
Type the name of the stream:
BioSemi
Type the uid of the stream:
EEG
Type the type of the stream:
f27926a2-6814-40e9-96be-3af52b6cfc63
Finding stream...
```

Fig. 3 Example of user input for select stream

If no such streams are found, a message will be printed on the terminal. If one stream is found, the program will immediately connect to the stream and a confirmation message will be printed on the terminal. If multiple streams of such type are found, the user is asked to select the stream they wish to connect to, after which the program connects to the selected stream.

```
Finding stream...
2024-11-24 16:13:45.820 (  28.195s) [        4A20D6B2]      netinterfaces.cpp:36    INFO|
2024-11-24 16:13:45.821 (  28.195s) [        4A20D6B2]      netinterfaces.cpp:36    INFO|
2024-11-24 16:13:45.821 (  28.195s) [        4A20D6B2]      netinterfaces.cpp:36    INFO|
2024-11-24 16:13:45.821 (  28.195s) [        4A20D6B2]      netinterfaces.cpp:36    INFO|
2024-11-24 16:13:45.821 (  28.195s) [        4A20D6B2]      netinterfaces.cpp:36    INFO|
2024-11-24 16:13:45.821 (  28.195s) [        4A20D6B2]      netinterfaces.cpp:36    INFO|
2024-11-24 16:13:45.821 (  28.195s) [        4A20D6B2]      netinterfaces.cpp:58    INFO|
2024-11-24 16:13:45.821 (  28.195s) [        4A20D6B2]         api_config.cpp:270   INFO|
2024-11-24 16:13:46.323 (  28.697s) [        4A20D6B2]             common.cpp:65    INFO|
Connected to stream
```

Fig. 4 Example output of a successful connection to a stream

3) Plot data

This program allows the user to connect to a stream and plots raw data from the stream. We have chosen to use the pyqtgraph graphics library to plot the data.

The program first searches for available streams. If none are found, a message will be printed. Otherwise, the user is asked to choose the stream to plot.

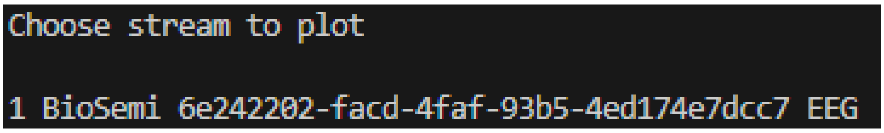

Fig. 5 Example output of available streams

Afterwards, the created inlet pulls data in chunks and plots the data onto a graph. Data from each channel is plotted on a separate curve. Each curve has a unique color (if the channel count is 8 or less).

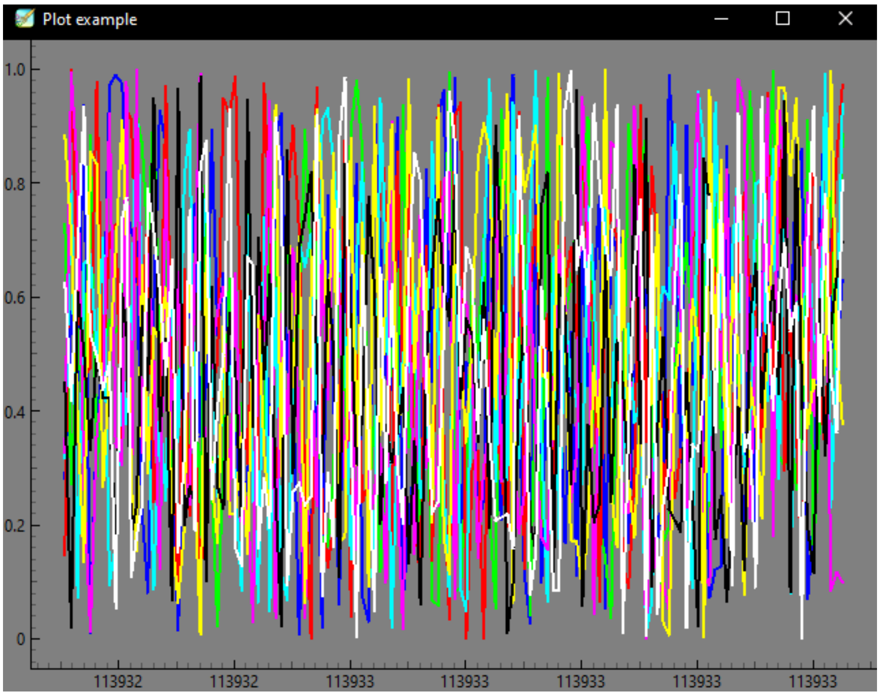

Fig. 6 Graph output for stream data

IV. Future expansion of examples

Future expansion plans include enhancing data visualization capabilities by allowing users to selectively display specific streams or transitioning to a stand-alone online viewer for more flexible and intuitive data analysis. Additionally, there will be developed new example focused on the filtration and analysis of biosignal data, further broadening the range of tools available for advanced data processing and insights. These updates aim to improve user experience, increase analytical flexibility, and support more sophisticated data handling in complex research environments[6].

V. Conclusion

The Lab Streaming Layer and Biosignals Laboratory provide powerful and versatile solutions for the collection, synchronization, and analysis of biosignal data. These tools streamline workflows by offering real-time data access, seamless integration with various devices, and simplified data management through configurable scripts. By alleviating the technical burdens of data management, LSL and BioLab empower researchers to delve deeper into analytical exploration, driving innovation and advancing discoveries in neuroscience and beyond. As these technologies evolve, they hold significant potential for further improving the efficiency and flexibility of biosignal research, paving the way for new discoveries and advancements in scientific exploration[1][5][6].

References

[1] C. Kothe, D. Medine, C. Boulay, M. Grivich, and T. Stenner, "Introduction — Labstreaminglayer 1.13 documentation," 2019. [Online]. Available: https://labstreaminglayer.readthedocs.io/info/intro.html. [Accessed: Nov. 24, 2024].

[2] C. Kothe, D. Medine, C. Boulay, M. Grivich, and T. Stenner, "User's Guide — Labstreaminglayer 1.13 documentation," 2019. [Online]. Available: https://labstreaminglayer.readthedocs.io/info/user_guide.html. [Accessed: Nov. 24, 2024].

[3] C. Kothe, D. Medine, C. Boulay, M. Grivich, and T. Stenner, "Supported Devices and Tools — Labstreaminglayer 1.13 documentation," 2019. [Online]. Available: https://labstreaminglayer.readthedocs.io/info/supported_devices.html. [Accessed: Nov. 24, 2024].

[4] "Tips and tricks for using Lab Streaming Layer (LSL)." [Online]. Available: https://www.brainproducts.com/support-resources/tips-and-tricks-for-lsl/. [Accessed: Dec. 1, 2024].

[5] C. Kothe, S. Y. Shirazi, T. Stenner, D. Medine, C. Boulay, M. Grivich, T. Mullen, A. Delorme, and S. Makeig, "The Lab Streaming Layer for synchronized multimodal recording," *bioRxiv*, 2024. [Online]. Available: https://doi.org/10.1101/2024.02.13.580071. [Accessed: Dec. 1, 2024].

[6] Q. Wang, Q. Zhang, W. Sun, C. Boulay, K. Kim, and R. L. Barmaki, "A scoping review of the use of Lab Streaming Layer framework in virtual and augmented reality research," *Virtual Reality*, 2023. [Online]. Available: https://doi.org/10.1007/s10055-023-00799-8. [Accessed: Dec. 1, 2024].

