Scientific Visualization, 2022, volume 14, number 4, pages 83 - 96, DOI: 10.26583/sv.14.4.08

How to Improve Utilizing Neural Interface by Means of Ontology-Driven Scientific Visualization Tools

S.I. Chuprina[1], I.A. Labutin[2]

Perm State University

[1] ORCID: 0000-0002-2103-3771, chuprinas@inbox.ru
[2] ORCID: 0000-0001-6858-1479, i.a.labutin@yandex.ru

Abstract

The technological progress in the field of Brain-Computer Interface and its integration with IoT have now put on the agenda the question of the fast transition of the technology from laboratory experiments into everyday life. The paper presents an approach to improve utilizing neural interface with the help of ontology-driven scientific visualization tools taking into account the urgent problems of automatic adaptation to the specifics of different IoT infrastructure, models and datasets.

Some issues of replicability and reproducibility of experiments are also under discussion in this paper. Using the principles of “clean-room reverse engineering” methodology to rewrite existing EEG device drivers we make it possible to embed visualization tools which dynamically render the streaming data coming from different EEG devices within a diverse IoT infrastructure without any legal complications.

Keywords: Internet of Things, Ontology Engineering, Brain-Computer Interface, Clean-Room Reverse Engineering, Replicability, Reproducibility.

1. Introduction

Despite significant progress in Brain-Computer Interfaces (BCI), many issues associated with collecting, analyzing and rendering Electroencephalography (EEG) signals in real-world environments still remain unresolved. This situation makes it difficult for researchers to use BCIs in multidisciplinary projects. Analyzing EEG data can get quite challenging. Signal processing, artifact detection and attenuation of undesired artifacts, feature extraction, and computation of mental metrics all require a high level of expertise and experience of researches to properly interpret and extract valuable information from the collected data.

As our experience tells us and our study has shown, there is currently a deficiency of high-level tools in the field of BCI that simplify the researchers’ work in the process of conducting experiments and improve visual data analysis, as is the case in Big Data technologies. In particular, there are no smart assistants helping the so-called Data Citizen to perform analytical work at the level of a qualified IT specialist. Also, it is crucial not only to utilize the best practices for neuroimaging but also make them based on a detailed discussion of different levels of repeatability, replicability and reproducibility.

These problems are directly related to another important issue concerning the methods of integration and adaptation of the developed visual tools to the third-party infrastructure of the Internet of Things, including BCI. Data, whether raw, processed, or segmented, can of course be exported to easily portable formats that allows visual analysis on any platform that researchers prefer. But real everyday practice shows that it is more effective and in demand to embed visualization tools directly into the scenario of the experiment being conducted. However, at the same time, third-party license agreements may be violated and the question arises of applying modern approaches, in particular, the so-called "clean room" method to solving this problem.

This paper is devoted to the description of approaches to solving the problems mentioned above.

2. Key contributions

Main focus points of this paper are devoted to tackling the following problems in order to improve the applicability of scientific visualization tools of SciVi developed by our team previously [1, 2, 3, 4] in BCI studies to adapt them to the specific of the different IoT infrastructure, models and datasets:

1. Decoupling the headset information in a form of ontology from the physical signal flow and making it available to other blocks of the pipeline as a separate data flow.
2. Creating a "sliding window" pipeline block that'll allow us to train and employ our classifier on chunks of data in an on-line manner.
3. Manner of using the principles of "clean-room reverse engineering" methodology to rewrite existing EEG device drivers to adapt them to the specific of the infrastructure of our experiments.

3. Related Work

In recent years, the significance of the integration of neural interfaces into an IoT ecosystem has become more and more clear and widely addressed in the literature. However, the problem of unified integration, despite being generally recognized, is still not considered wide enough [5, 6]. Nishimura et al. proposed a system called BIRT based on XML configuration that allows easy adjusting and modifying experiments' pipeline [7]. Quitadamo et al. described a custom BCI communication protocol using UML notation [8]. Camelo et al. applied genetic algorithms to the task of controlling the smart conference room with commercial-grade BCI [9]. Mendez et al. created an ontology describing BCI communication with nodes of IoT infrastructure [10, 11]. Zao et al. proposed the concept of "an augmented BCI" or A-BCI [12]. They applied ontological engineering to create an ontological description of a fog ecosystem, and demonstrated their approach using a BCI-controlled online game

The problem of replicability of the results of EEG-based studies is also a major topic in the neuroscientific community. The "EEGManyLabs" project [13] aims to replicate some influential EEG-based neuro experiments in order to confirm their studies and provides a set of recommendations to the future researchers about conducting their experiments in a reproducible way. The Organization for Human Brain Mapping is also raising awareness of this problem and has developed a guideline for any neuroscientist to follow in their work [14]. Detailed information about the equipment used in research is thus very important, and so are the software tools. Employment of the open-source software [15] and open communication protocols, such as VSCP (https://vscp.org) or LSL (https://github.com/sccn/labstreaminglayer), allows any researcher not only to replicate the experiments' pipeline and validate results achieved, but also to use it as a baseline for further extension [16].

While offline classification and data processing is of a very major importance in neuroscience, the significance of online (streaming, real-time) data processing can't be underestimated. To conduct many types of studies we might be content with offline processing of the prerecorded data; however, if we want to research BCI, especially, in the field of HCI, we're bound to require some kind of online algorithms. It's also very important in medical neuroscience studies, as early seizure detection, for example, allows timely medical care and more thorough analysis of occurring phenomena [17]. However, streaming data comes with its own set of hardships. In [18] the data streams are defined as a dynamic set of data where:

- Elements of the data arrive in the real-time.
- System has no effective control over the order of the elements.
- There's potentially no limit on the number of elements of the data.

- After processing elements of the data in some way, they are either archived or discarded, and the number of discarded elements is prevalent over the archived ones.

Authors of [19] emphasize four major aspects of streaming data: Volume, Variety, Velocity and Volatility, out of which two latter ones are the most important to distinguish streaming data from other data sources. Velocity of streaming data makes it hard to process, while Volatility is complicating hypothesizing about the data.

Several types of problems related to visualization in the experiments with streaming data are distinguished in [19]:

- Context Preservation. It's very important for analytic tools to store information about the past and provide a convenient way to recover it when the need arises.
- Mental Map Preservation. Changes in streamed data might be rapid and complex, therefore a visualizer should present them in a way that allows a human researcher to recognize patterns and react to them.
- Change-Blindness Prevention. A human observer can look at the visualization for a long period of time and his/her eyes might start to glaze over the changes in data. A good visualization should help person with that.
- Time representation. Temporal dimension is of utmost importance when analyzing streaming data, especially in neuroscience, and it's not always clear how it should be presented to the researcher.

Main types of visualization used in neuroscience research papers are:

- Raw signal data (electromagnetic potentials or oxygen levels) [20, 21] allows a trained professional to spot a phenomenon or pattern in the most direct way possible. It is usually right after the "Receiving biosignal" step in the experimental setup (Fig. 1).
- Spatial locations of signals w.r.t a head [22, 23, 24] provide information about neural activity localization in specific parts of the brain. They could be shown on a realistic 3D brain model or just on a simplistic head visual scheme, and are usually used on the "Feature extraction" step (Fig. 1) to help a researcher get a better hold of occurring phenomena and choose appropriate feature selection algorithm.
- In case of EEG studies, signal spectrum [25, 26, 27] is used, because different EEG phenomenon has distinct spectral characteristics which can be used to distinguish such phenomennon. Visualizing it helps on "Feature extraction" step of the experiment too (see Fig. 1).

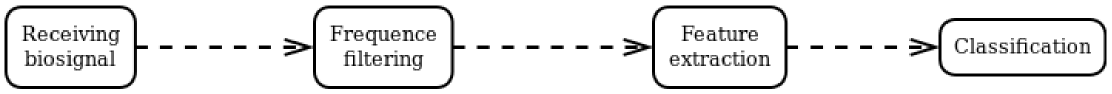


Fig. 1. A common scheme of neuroscience experiment involving EEG data processing

Thanks to the comprehensive review [28] we can be free from going into a detailed survey of current challenges and opportunities of modern BCI. However, there are no accessible published findings related to the methods of adaptable embedding of visualization tools into third-party infrastructure integrating BCI and IoT.

As a part of our previous work, we built and successfully tested a pipeline that utilizes a unified high-level mechanism to allow brain-computer interfaces to be integrated into diverse IoT ecosystems in a way that doesn't depend on the characteristics of a particular ecosystem [4]. That pipeline was developed using the SciVi platform, an ontology-driven scientific visualization and visual analytics toolset [1]. Then we extended this pipeline to apply it for the task of audiovisual stimuli presentation for the neurophysiological studies [2, 3]. However, there were two major technical issues with this pipeline implementation that we're going to address in this paper:

1. As a part of this pipeline we utilized an EEG device driver with a closed-source code we weren't able to publish. This severely limited the ability to reproduce our results by independent researchers, thus making it not quite aligned with moderns' scientific standards. We tackle this problem by applying a "clean room" reverse engineering methodology to reproduce driver's functionality in a new, clean reimplementation we can share with a community.

While we were at it, we also moved some logic related to labeling the signal components out of the driver to a new pipeline node.

2. Our unified pipeline was split into two separate parts, namely, the recording part and the processing part. To conduct a study, we had to perform a full data recording cycle first and only then analyze the acquired data in an offline manner. This reduced the range of available experiment setups we were open to, and also occluded our ability to react to the experiment's conditions (e.g., we'd only be able to notice something went wrong post factum, after a full recording cycle had been completed). So, we add new nodes to a pipeline to make online processing of streaming data possible.

4. Implementation

4.1. Decoupling Headset Information from Signal

In the previous iteration of our pipeline, the information about a headset – electrodes, their names and positions, mapping from the electrodes to EEG channels etc. – were provided by the "EBNeuro" node and baked together with signal flow. Every block receiving signal data was also receiving headset information (even if it didn't require it), and every block that required information about a headset had to subscribe to a signal stream. That was not only more resource-consuming than necessary but in fact led us to some duplication of the work (e.g., "Impedance visualization" block had to rely on externally provided headset information due to technical complications).

Now we extracted all information related to the headset into a separate block called "Montage Provider". It's supplied with ontological description of the headset and can be used as a convenient source that incapsulates all the necessary information about the electrodes and their properties in one place and is accessible from any other pipeline blocks. Fig. 2 shows an example of such ontological description used in our experiments (21-channel EBNeuro headset montage). As usual, we restrict the set of relation types used in the ontology only by paradigmatic types ("is a", "has", "is used for") in order to reduce the complexity of the reasoner allowing to embed it to Edge devices as firmware [4], which ensures semantic interoperability within IoT networks. And we used visual editor ONTOLIS to create ontologies.

As of now we only employ one montage in our experiments, but as the system grows it might be necessary to create some comprehensive way of storing and managing montages. This could be achieved with the help of some kind of high-level smart repository and is a topic of a future work.

Comparison of the old pipelines (a) and the new ones (b) is shown on Fig. 3. This separation also allows us to reuse "Montage Provider" node at the stage of configuring experimental setup.

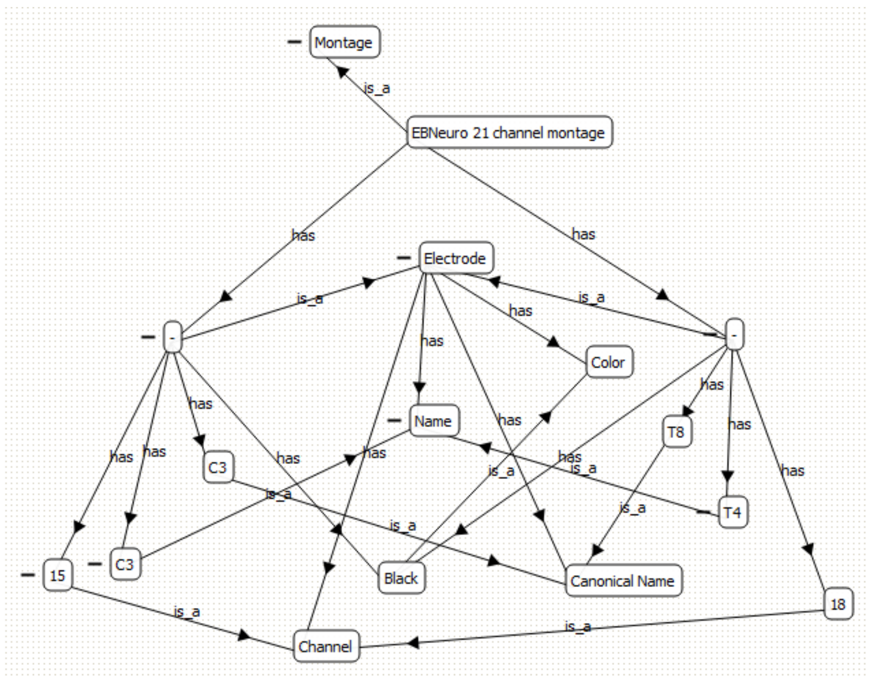
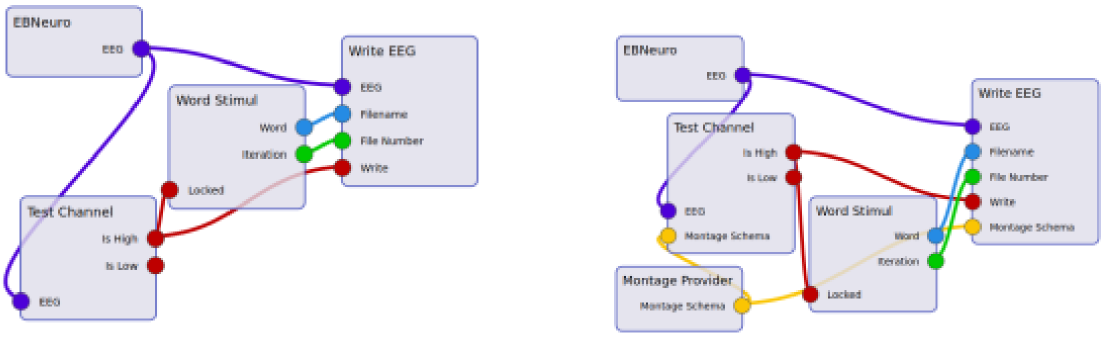


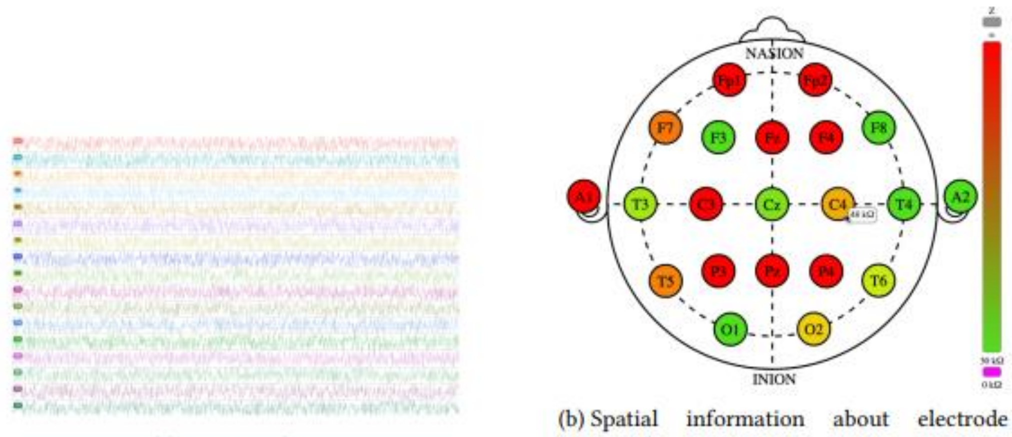
Fig. 2. Ontology example for "Montage Provider" pipeline node (only two out of 21 electrodes are shown for clarity)



(a) Old pipeline

(b) New pipeline with "Montage Provider" node

Fig. 3. Side-by-side comparison of pipelines



(a) EEG raw data

(b) Spatial information about electrode impedances

Fig. 4. An example of visual representations for monitoring different pipeline attributes [2, 3]

description which allows for quick swapping of different headsets in the experiment.

In our previous paper [2, 3] we demonstrated a data monitoring pipeline (Fig. 4), but as of time of writing it had some drawbacks; in particular, visualization of the headsets' impedances was static in reference to electrode counts and their positions.

New "Montage Provider" node allows us to provide a unified way to feed headset information into a pipeline. Now to visualize impedances we can use a headset ontological

4.2. Sliding Window for Data Streaming

During our previous experiments, we were limited to offline classification and processing due to the different organization of two pipelines: signal acquisition pipeline operated frames of data, while processing and classification pipeline utilized groups of frames. We introduce a new block called "Sliding window" that uses a simple idea of accumulating last N frames and outputting them together as a group (N is a configuration parameter of this block). Thus, we are now able to simultaneously acquire the signal, process it and train/evaluate our classifier in a streaming manner. Ontology describing this node is illustrated on Fig. 5.

We also were required to introduce another new node – "Labels by Channel" – for labeling such frames based on the value of some channel present in data. Previously in offline processing this role was fulfilled by "MNE-EEG Converter" node, but the combining of two functions into the same block allowed only the processing of a prerecorded data that was loaded in MNE format. New node functionally quite similar to the previously existing "Test Channel" node except it doesn't buffer data as this role is now performed by "Sliding Window". "Labels by Channel" node computes some integral characteristics (there're quite a few to choose from – mean, median, mode etc.) of a specific channels' signal over an entire frame and outputs "1" as a label if this value is above specified threshold or "0" in the other case.

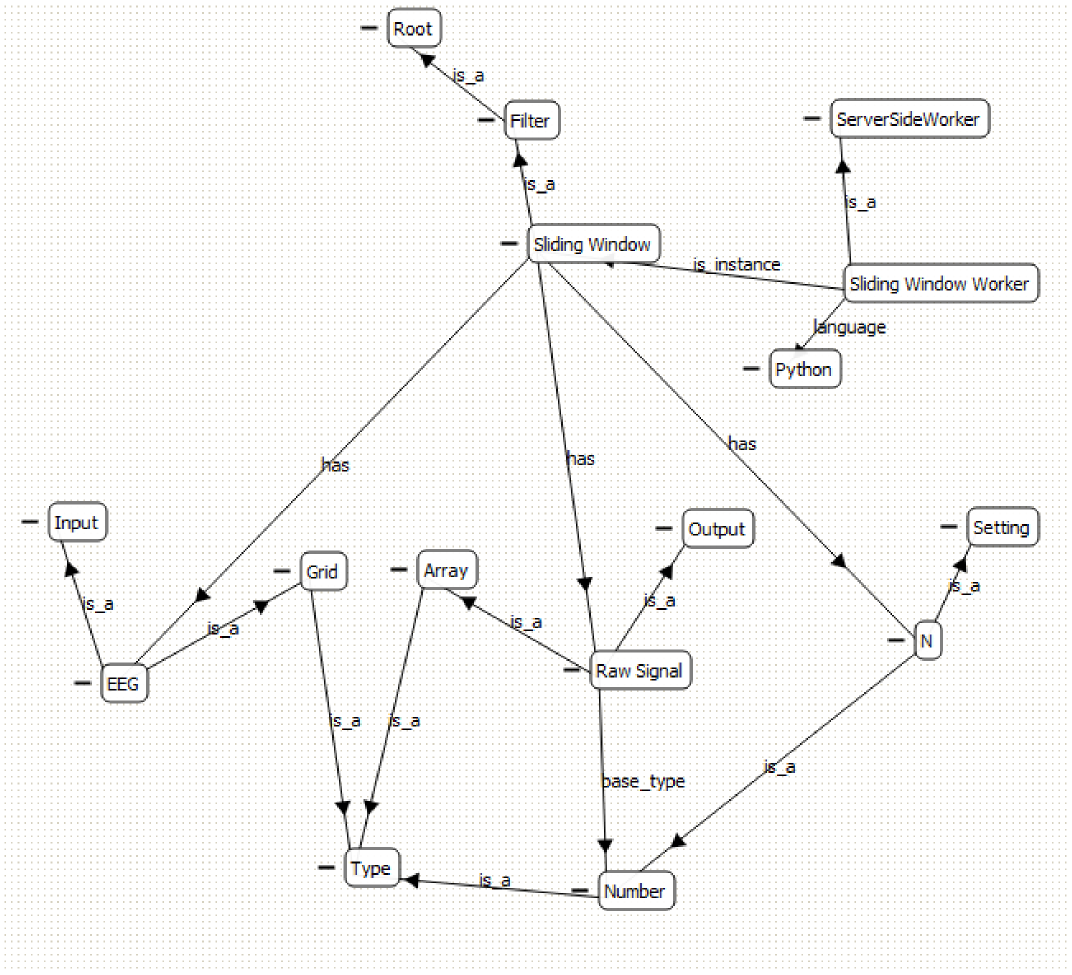
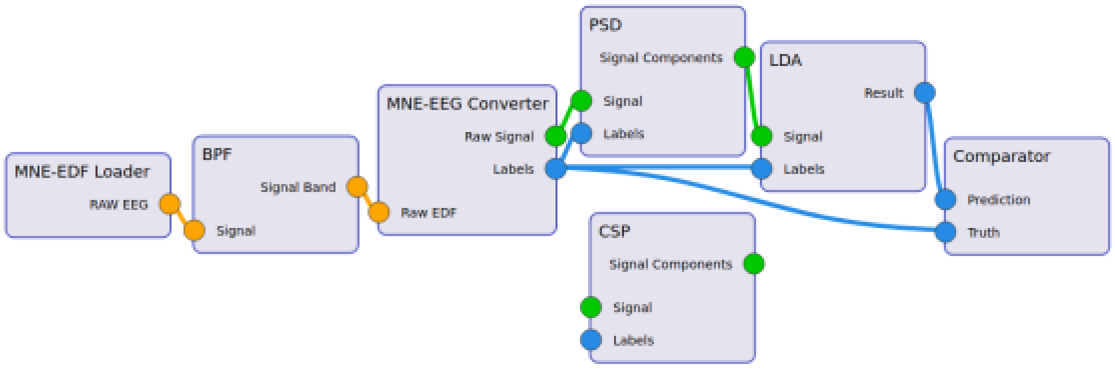
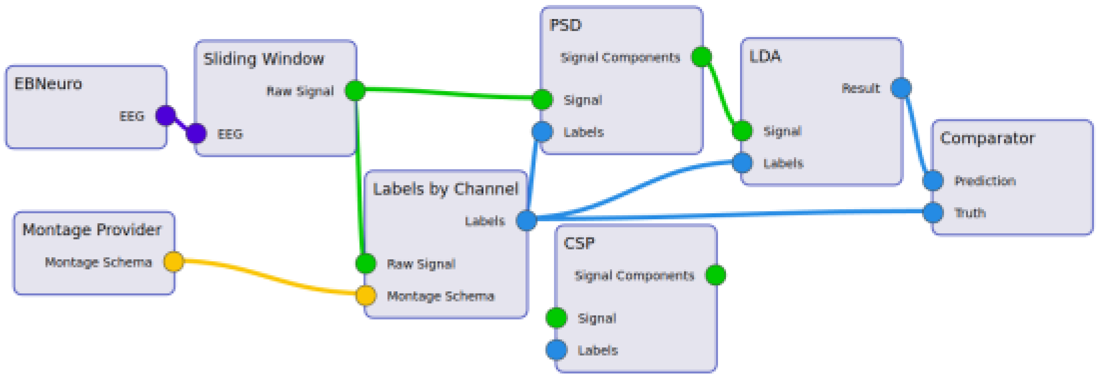


Fig. 5. Ontology for "Sliding Window" node

Comparison of the old pipeline without new nodes (top) and a new one with them (bottom) is presented on Fig. 6. Beware that the old pipeline was offline processed and relied on the data prerecorded with a pipeline similar to the one presented on Fig. 3, while the new one enables online processing of the data in a streaming manner. CSP node is shown but isn't connected to the pipeline; it can be used in the pipeline instead of PSD node and just added only for illustration.



(a) Old offline data processing pipeline



(b) New online data processing pipeline

Fig. 6. Pipelines without (a) and with (b) sliding window

4.3. Clean-Room Reverse Engineering of the EEG Driver

E. J. Chikofsky and J.H. Cross in [29] define the term "Reverse engineering" as the process of analyzing the target system in order to determine its components and their interaction, and then creating some high-level description of this system. The authors also define the term "Reengineering" as a re-creation of the target system in some new form. Reengineering involves the reverse engineering of the target system and the subsequent "forward" development based on the acquired knowledge, including any changes to the new system if necessary. Thus, reengineering is a re-creation of some product based on information obtained through reverse engineering.

It is easy to imagine that a researcher using the results of reverse engineering can run into legal implications at some point. The author of [30] notes that a program created as a result of reworking some other program becomes a derivative work. So, the rights of the author of the original program are still in effect. The article provides a clarification: a similarly functioning program cannot be declared copied unless the real use of the original code or parts of it is proven.

To avoid such copying, there are several approaches to reverse engineering, but the simplest and most popular is clean room reverse engineering. It's based on division of labor to two teams:

1. The research team uses all the reverse engineering methods and tools available to create a technical documentation describing structure and behavior of the target system. All implementation details that may be considered an intellectual property are excluded from this documentation.
2. Next, the development team, consisting exclusively of people who did not participate in the previous step, writes a new program based on this documentation.

This approach eliminates copying parts of the code and avoids any legal implications. It is sufficiently fast, as it does not limit the possible tools used for research, but at the same time requires a separate team of people for creating the technical documentation who cannot participate in the development of a new system in any way. During the initial design of the driver, an architectural decision was made that the driver should be implemented as a single class that directly reflects the functionality of the device. We also separated the platform-specific code with a special abstraction.

This abstraction hides the operating system on which the driver is running (in the case of microcontrollers, this abstraction can play the role of the operating system itself) and includes networking and debug output. The interface of an abstract data type was declared in a C header file that defines some types and functions (system methods). An implementation of this interface was made targeting operating systems based on the Linux kernel.

The implemented basic functionality of the device is as follows:

- To initialize the device, the corresponding method connects to the device's initialization port, requests service information, puts the control and data ports into the ready state and then connects to them.
- To receive data from the device, a method is implemented that reads packets from the data port into a special queue, from which measurements are transmitted to the user. The use of a queue avoids data loss during slow processing.
- To de-initialize, the EEG is put into standby mode, after which the control and data ports are closed and disabled.

While the created driver can be used as a standalone library, in our experiments we are interested in integrating it into an existing pipeline as a block. SciVi platform supports several ways of integration modules into its environment, but the most uniform way is creating a Python wrapper module and its specification as ontology. Thus, a Python wrapper for the driver was implemented using the SWIG tool.

This approach enables us to embed visualization tools which dynamically render the streaming data coming from different EEG devices within a diverse IoT infrastructure without any legal complications. The project uses CMake [31] as a build system. Source code is available under GPLv3 license at https://github.com/icosaeder/libmed. Protocol documentation can be supplied upon further request.

5. Using Scientific Visualization Tools to Improve Experiments

Approaches and visualization methods described in [26, 27, 31] (see, for example, Fig. 7) are of great interest to apply in the pipeline of our experiments. All of these methods and tools should be investigated and, if possible, reused and compared to improve the quality of the current version of our toolset and support the researcher through their decision-making processes in more comprehensive way. Using the different scientific visualization tools for analysis the results of EEG-based experiments with the same dataset can help not only choosing more adequate parameters for some machine learning algorithm, but also the replacement one clustering or classifying algorithm with another, if the results of visual analysis show that it is reasonable.

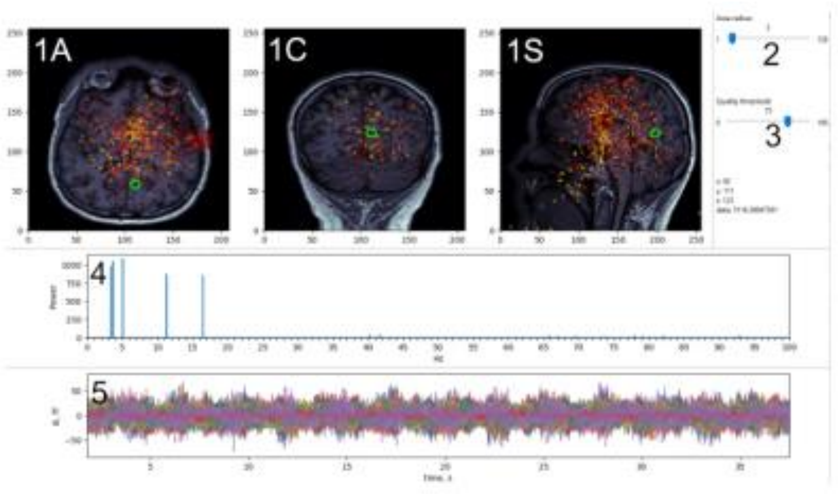
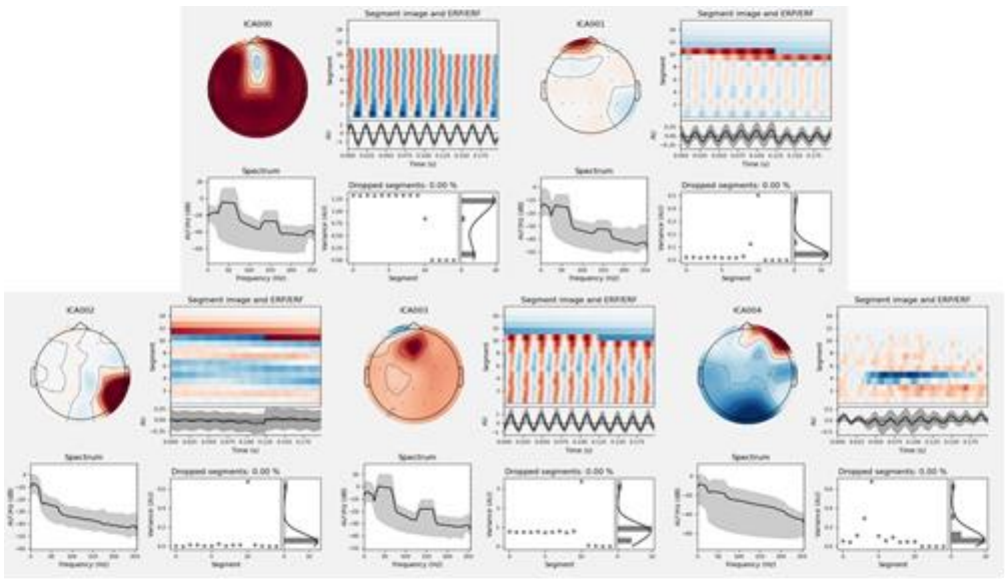


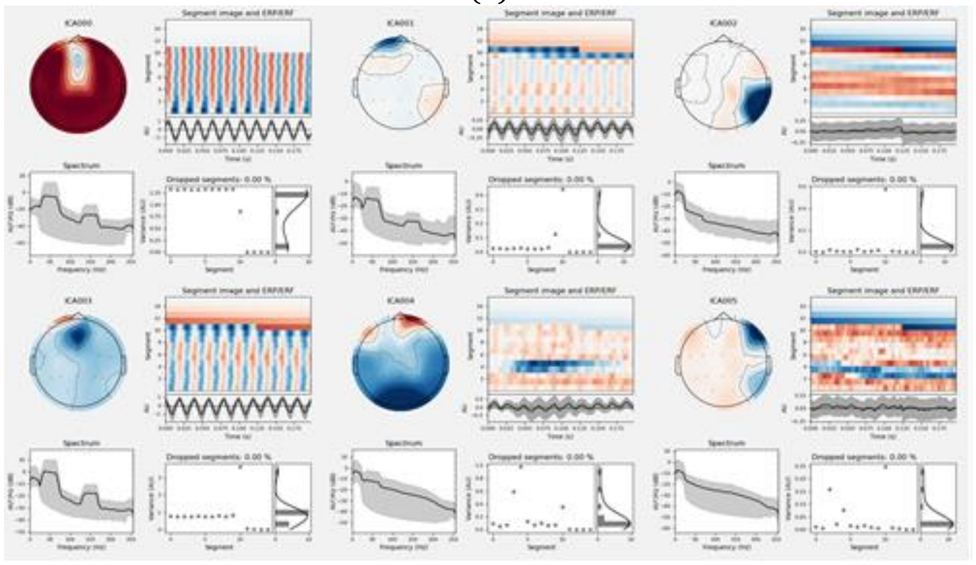
Fig. 7. Pipelines MEG visualization example [27]

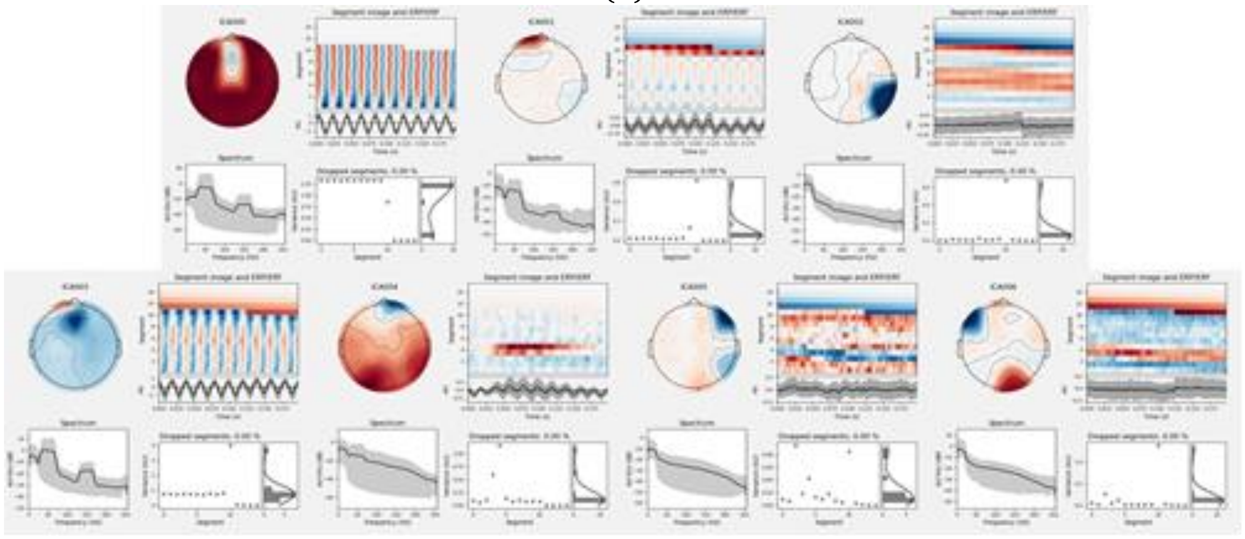
The raw signal itself already allows trained professionals to grasp some properties of data, so it definitely should be one of the primary targets for online visualization. It also makes it possible to detect some discrepancies in the setup (noisy electrode, static/linear noise, etc.). Filtered signal should be visualized too, as it excludes a possible filtering error dealing with removing some channel or frequency band, e.g. We have already demonstrated the result of our implementation of this feature above (see Fig. 4). Visualization of brain activity zones in real-time akin to the impedance visualization at Fig. 4 is very useful in the studies related to motor imagery or other relatively localized brain artifacts.

In our study we use visualization of factorization outcome that is also vital as it allows us to better understand how the factorization algorithm performed and how it may help to improve the results of the study. Visualization of ICA results (Fig. 8) help us to analyze the quality of extracted components, adjust the hyperparameters and therefore improve the classification results. Each component of visualization is presented in 4 quadrants: top left – brain activity topography; top right – event-related brain activity by segments; bottom left – signal spectrum; bottom right – signal variance.

On Fig. 11 we can see peaks on 50, 150 and 250 Hz frequencies in some components' spectrum being visualized, which may indicate that there is no online noise filter. We can also compare the quality of the factorization in different experiments where the same dataset is used but the given number of components is different. In our case we can hypothesize that according to visualization of the segmentation results and variance increasing the number of components improves the quality of method, so it may be beneficial to choose a larger value for this parameter.



(a)


(b)


(c)

Fig. 8. ICA test run for different number of components: ICA for 5 components (a), ICA for 6 components (b), ICA for 7 components (c)

Another useful visualization is plotting accuracy and other metrics of different classification algorithms as they process the data. For this, we suggest use the Scikit-learn (sklearn) that is one of the most widely used Python packages for Data Science and Machine Learning [32].

6. Conclusions

This paper deals with different issues that make it possible to automate the research conduction by means of embedding the ontology-driven scientific visualization tools in third party infrastructure IoT that include BCI. Also, to improve utilizing neural interface we take into account the replicability and reproducibility issues and suggest the manner of using the principles of clean-room reverse engineering methodology to rewrite existing EEG device drivers that help us to reproduce the experiments without any legal complications.

Now together with the researchers from the educational and scientific laboratory of socio cognitive and computational linguistics of Perm State University we have successfully used the suggested solutions to automate the experiments on the analysis of human perception of visual incentives (adjectives of the Russian language) using EEG. Two cases of perception of adjectives were considered, based on a comparison of visual and auditory perceptual modalities and on a comparison of primary and secondary formed perceptual modalities.

In future, we plan to expand the study of different visual methods, which can be utilized in EEG-based projects, to enrich the SciVi repository and tools to adapt them to utilizing BCI within different IoT infrastructure and simplify researcher's decision making process, in particular, by means of choosing adequate machine learning methods taking into account the nature and specifics of analyzed datasets.

7. Acknowledgements

The study has been performed as a part of the research project “State task of Perm State University for 2020-2022”, topic number FSNF-2020-0023 “Conceptualization of social reality in mass communication: cognitive-information modeling using machine learning methods, visual analytics and neurocognitive technologies”.

1. K. Ryabinin, S. Chuprina, High-level toolset for comprehensive visual data analysis and model validation, Procedia Computer Science 108 (2017), pp. 2090–2099. URL: https://www.sciencedirect.com/science/article/pii/S1877050917305690. doi:10. 1016/j.procs.2017. 05.050, international Conference on Computational Science, ICCS 2017, pp. 12-14. June 2017, Zurich, Switzerland.

2. K. Ryabinin, S. Chuprina, I. Labutin, Ontology-driven toolset for audio-visual stimuli representation in EEG-based BCI research, in: Proceedings of the International Conference on Computer Graphics and Vision “Graphicon”, CEUR, volume 31, Keldysh Institute of Applied Mathematics, 2021, pp. 223–234. URL: https://keldysh.ru/papers/2021/prep_vw.asp?pid=9273&lg=e. doi:10.20948/graphicon-2021-3027-223-234. arXiv:http://ceur-ws.org/Vol-3027/paper21.pdf.

3. K. V. Ryabinin, S. I. Chuprina, I. A. Labutin, Ontology-driven tools for EEG-based neurophysiological research automation, Scientific Visualization 13.4 (2021), pp. 93–110. doi:10.26583/sv.13.4.08.

4. K. V. Ryabinin, S. I. Chuprina, I. A. Labutin, Tackling IoT interoperability problems with ontology-driven smart approach, Lecture Notes in Networks and Systems 342 (2021), pp. 77–91. doi:10.1007/978-3-030-89477-1_9.

5. B. Allison, The I of BCIs: Next generation interfaces for brain-computer interface systems that adapt to individual users, in: J. A. Jacko (Ed.), Human-Computer Interaction. Novel Interaction Methods and Techniques, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp.◦558–568.

6. S. Huang, E. Tognoli, Brainware: Synergizing soffiware systems and neural inputs, in: Companion Proceedings of the 36th International Conference on Soffiware Engineering, ICSE Companion 2014, Association for Computing Machinery, New York, NY, USA, 2014, p.p. 444–447. URL: https://doi.org/10.1145/2591062.2591131. doi:10.1145/2591062.2591131.

7. E. M. Nishimura, E. D. Rapoport, P. M. Wubbels, T. H. Downs, J. H. Downs, Functional Near-Infrared Sensing (fNIR) and Environmental Control Applications, Springer London, London, 2010, pp. 121–132. URL: https://doi.org/10.1007/978-1-84996-272-8_8. doi:10.1007/ 978-1-84996-272-8_8.

8. L. R. Quitadamo, M. G. Marciani, G. C. Cardarilli, L. Bianchi, Describing different brain computer interface systems through a unique model: A UML implementation, Neuroinformatics 6 (2008), pp. 81–96. URL: https://doi.org/10.1007/s12021-008-9015-0.

9. G. A. Camelo, M. L. Menezes, A. P. Sant'Anna, R. M. Vicari, C. E. Pereira, Control of smart environments using brain computer interface based on genetic algorithm, in: N. T. Nguyen, B. Trawinski, H. Fujita, T.-P. Hong (Eds.), Intelligent Information and Database Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016, pp.◦773-781.

10. S. J. R. Mendez, J. K. Zao, BCI ontology: A context-based sense and actuation model for brain-computer interactions, in: SSN@ISWC, 2018.

11. S. Jose, R. Mendez, Modeling actuations in BCI-O: a context-based integration of SOSA and IoT-O, in: Proceedings of the 8th International Conference on the Internet of Things, IOT '18, Association for Computing Machinery, New York, NY, USA, 2018. URL: https: //doi.org/10.1145/3277593.3277914. doi:10.1145/3277593.3277914.

12. J. K. Zao, T. T. Gan, C. K. You, S. J. R. Mendez, C. E. Chung, Y. T. Wang, T. Mullen, T. P. Jung, Augmented brain computer interaction based on fog computing and linked data, in: 2014 International Conference on Intelligent Environments, 2014, pp.◦374–377. doi:10.1109/IE.2014.54.

13. Y. G. Pavlov, N. Adamian, S. Appelhoff, M. Arvaneh, C. S. Benwell, C. Beste, A. R. Bland, D. E. Bradford, F. Bublatzky, N. A. Busch, P. E. Clayson, D. Cruse, A. Czeszumski, A. Dreber, G. Dumas, B. Ehinger, G. Ganis, X. He, J. A. Hinojosa, C. Huber-Huber, M. Inzlicht, B. N. Jack, M. Johannesson, R. Jones, E. Kalenkovich, L. Kaltwasser, H. Karimi-Rouzbahani, A. Keil, P. Konig, L. Kouara, L. Kulke, C. D. Ladouceur, N. Langer, H. R. Liesefeld, D. Luque, ¨ A. MacNamara, L. Mudrik, M. Muthuraman, L. B. Neal, G. Nilsonne, G. Niso, S. Ocklenburg, R. Oostenveld, C. R. Pernet, G. Pourtois, M. Ruzzoli, S. M. Sass, A. Schaefer, M. Senderecka, J. S. Snyder, C. K. Tamnes, E. Tognoli, M. K. van Vugt, E. Verona, R. Vloeberghs, D. Welke, J. R. Wessel, I. Zakharov, F. Mushtaq, #eegmanylabs: Investigating the replicability of influential EEG experiments, Cortex 144 (2021), pp. 213–229. URL: https://www.sciencedirect.com/science/article/pii/S0010945221001106. doi:https://doi.org/10.1016/j.cortex.2021.03.013.

14. C. Pernet, M. I. Garrido, A. Gramfort, N. Maurits, C. M. Michel, E. Pang, R. Salmelin, J. M. Schoffelen, P. A. Valdes-Sosa, A. Puce, Issues and recommendations from the OHBM COBIDAS MEEG committee for reproducible EEG and MEG research, Nature Neuroscience 23 (2020), pp. 1473–1483. URL: https://doi.org/10.1038/s41593-020-00709-0.

15. A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, M. S. Matti Hämäläinen, MEG and EEG data analysis with MNE-Python, Frontiers in Neuroscience 7 (2013), pp.◦1–13. doi:10.3389/fnins.2013.00267.

16. J. Katona, A. Kovari, EEG-based computer control interface for brain-machine interaction, International Journal of Online Engineering (iJOE) 11 (2015), pp. 43–48. doi:10.3991/ijoe.v11i6. 5119.

17. R. Meier, H. Dittrich, A. Schulze-Bonhage, A. Aertsen, Detecting epileptic seizures in long-term human EEG: A new approach to automatic online and real-time detection and classification of polymorphic seizure patterns, Journal of Clinical Neurophysiology 25 (2008). URL:

https://journals.lww.com/clinicalneurophys/Fulltext/2008/06000/Detecting_ Epileptic_Seizures_in_Long_term_Human.1.aspx.

18. B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, Models and issues in data stream systems, in: Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 2002, pp. 1–16. doi:10.1145/543613.543615.

19. A. Dasgupta, D. L. Arendt, L. R. Franklin, P. C. Wong, K. A. Cook, Human factors in streaming data analysis: Challenges and opportunities for information visualization, Computer Graphics Forum 37 (2018), pp. 254–272. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13264. doi:https://doi.org/10.1111/cgf.13264. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13264.

20. L. Kangassalo, M. Spape, T. Ruotsalo, Neuroadaptive modelling for generating images matching perceptual categories, Scientific Reports 10 (2020) 14719. URL: https://doi.org/10. 1038/s41598-020-71287-1.

21. M. N. Ustinin, S. D. Rykunov, A. I. Boyko, E. F. Tarasov, I. V. Zhuravlev, M. A. Polikarpov, T. A. Ryabov, I. A. Filatov, A. Y. Yurenya, V. Y. Panchenko, The study of the perception of written speech by the method of functional tomography according to electroencephalography, Mathematical Biology and Bioinformatics 16 (2021), pp. 1–14. doi:10.17537/2021. 16.1. [in Russian]

22. S. Park, C.-H. Han, C.-H. Im, Design of wearable eeg devices specialized for passive brain-computer interface applications., Sensors (Basel, Switzerland) 20 (2020).

23. P. Bobrov, A. Frolov, C. Cantor, I. Fedulova, M. Bakhnyan, A. Zhavoronkov, Brain-computer interface based on generation of visual images, PLOS ONE 6 (2011), pp. 1–12. URL: https: //doi.org/10.1371/journal.pone.0020674. doi:10.1371/journal.pone.0020674.

24. V. V. Kozunov, A. Ossadtchi, Gala: group analysis leads to accuracy, a novel approach for solving the inverse problem in exploratory analysis of group meg recordings, Frontiers in Neuroscience 9 (2015). URL: https://www.frontiersin.org/articles/10.3389/fnins.2015.00107. doi:10.3389/fnins.2015.00107.

25. S. D. Rykunov, E. S. Oplachko, M. N. Ustinin, R. R. Llinas, Methods for magnetic encephalography data analysis in MathBrain cloud service, Mathematical Biology and Bioinformatics 12 (2017), pp. 176–185. doi:10.17537/2017.12.176.

26. M. N. Ustinin, S. D. Rykunov, A. I. Boyko, O. A. Maslova, Reconstruction of the functional structure of the human brain according to electroencephalography data, Mathematical Biology and Bioinformatics 15 (2020), pp. 106–117. doi:10.17537/2020.15.106. [in Russian]

27. S. D. Rykunov, E. D. Rykunova, A. I. Boyko, M. N. Ustinin, “VirtEl” software package for magnetic encephalography data analysis using the virtual electrode method, Mathematical Biology and Bioinformatics 14 (2019), pp. 340–354. doi:10.17537/2019.14.340. [in Russian]

28. S. Saha, K. A. Mamun, K. Ahmed, R. Mostafa, G. R. Naik, S. Darvishi, A. H. Khandoker, M. Baumert, Progress in brain computer interface: Challenges and opportunities, Frontiers in Systems Neuroscience 15 (2021). URL: https://www.frontiersin.org/articles/10.3389/ fnsys.2021.578875. doi:10.3389/fnsys.2021.578875.

29. E. Chikofsky, J. Cross, Reverse engineering and design recovery: a taxonomy, IEEE Soffiware 7 (1990), pp. 13–17. doi:10.1109/52.43044.

30. S. V. Sandalova, Legal features of derivative and composite computer programs, Bulletin of science and education 3 (2015), pp.◦141–146. URL: https://cyberleninka.ru/article/n/pravovye-osobennosti-proizvodnyh-i-sostavnyh-programm-dlya-evm. [in Russian]

31. K. Martin, B. Hoffman, Mastering CMake: A Cross-platform Build System, Kitware Incorporated, 2013. URL: https://books.google.ru/books?id=Zw3tvQEACAAJ.

32. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011), pp. 2825–2830.