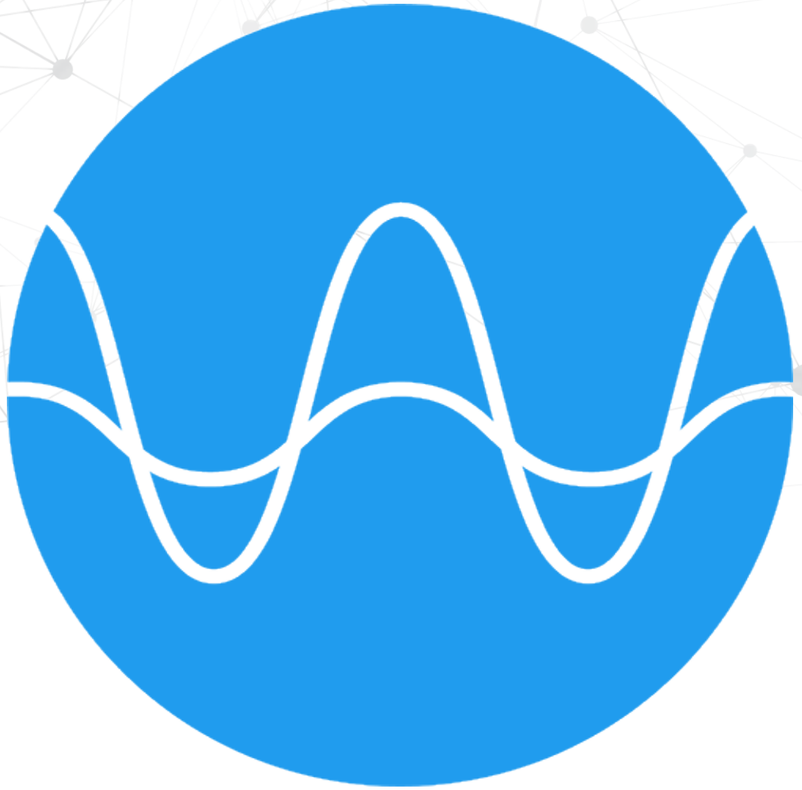
TIMEFLUX

OPEN-SOURCE PYTHON FRAMEWORK FOR BCI



Overview

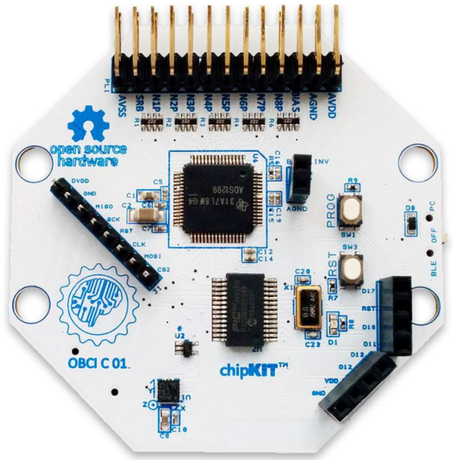
Use cases

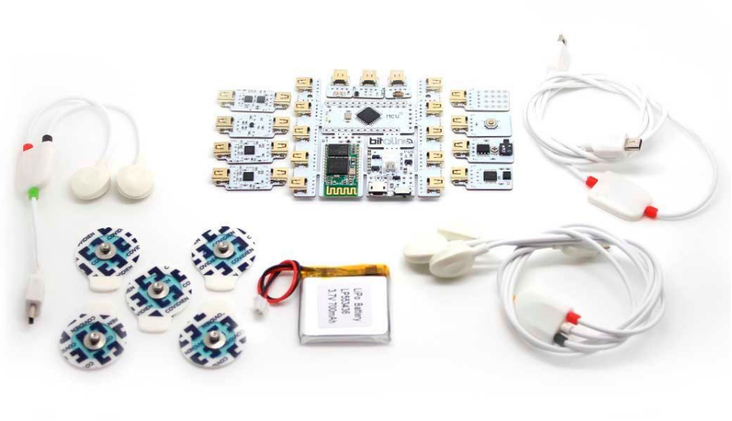
- Data and event **acquisition** from multiple sources
- **Stimulus** presentation
- Bio-feedback
- **Brain-Computer Interfaces**
- Interactive installations
- And more :)

In a nutshell

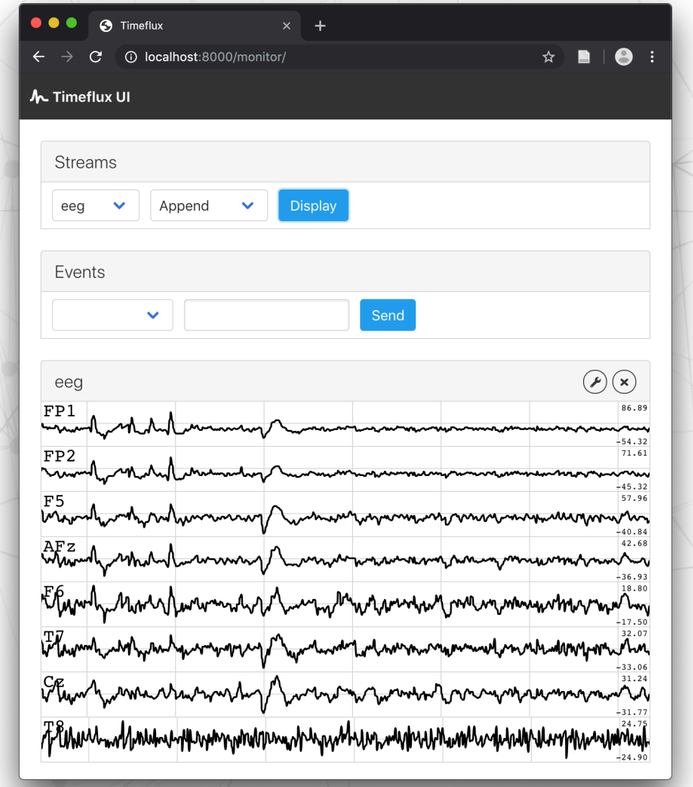
- Fits well within the **Python datascience** ecosystem
- Permissive **MIT license**: commercial use authorized
- **Independent** project funded through consulting services
- Works both **offline** and **online**
- Quick **prototyping**

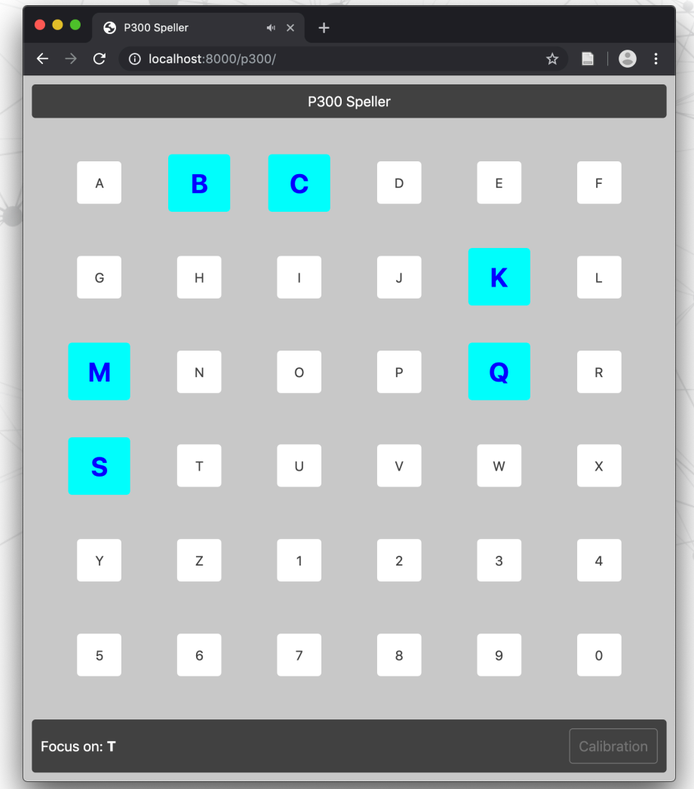
It works with your devices

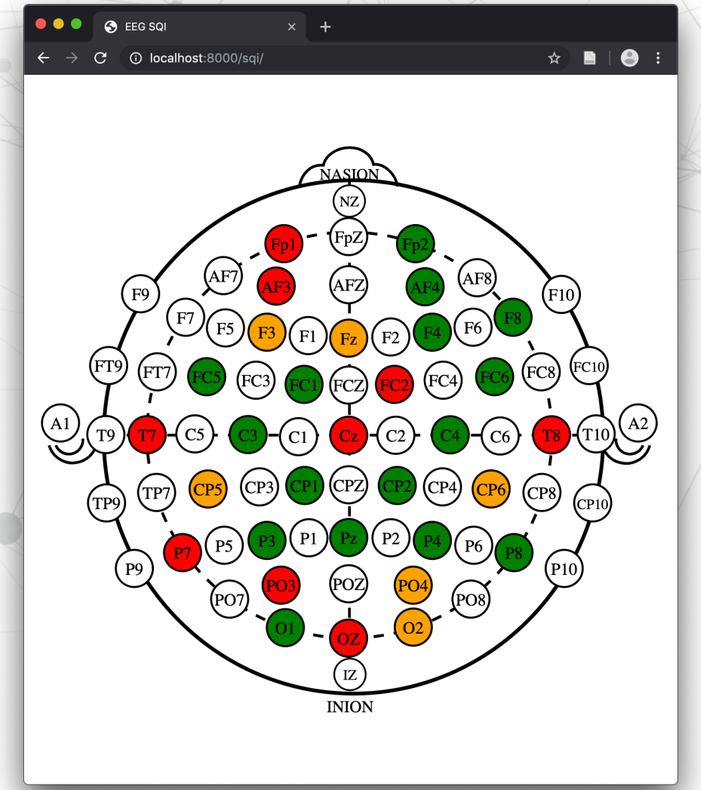




It happens in the browser







Easy to learn, use, and extend

- **Familiar concepts**: graphs, nodes, edges
- Relies on **industry standards**: Pandas, Xarray, Scikit-Learn, Lab Streaming Layer
- **Descriptive pipelines**: simple YAML syntax, no coding required
- **Custom nodes**: standard Python classes

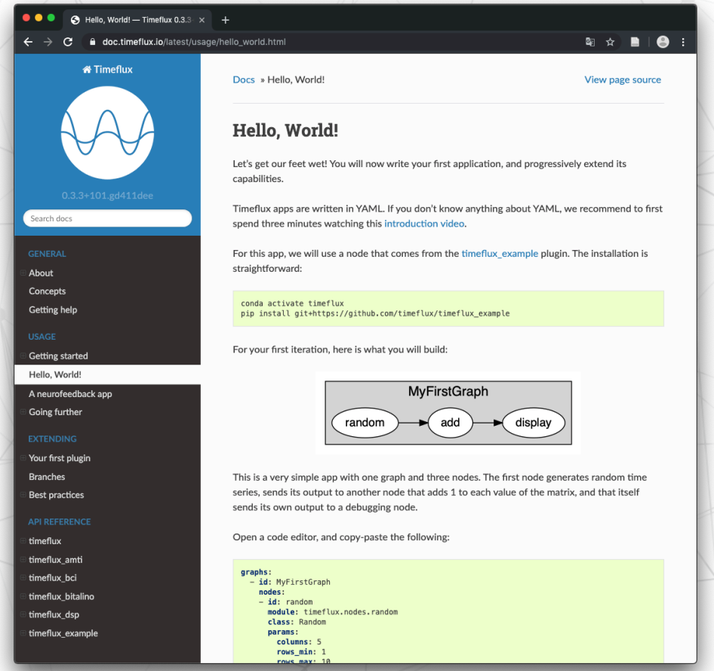
Batteries included

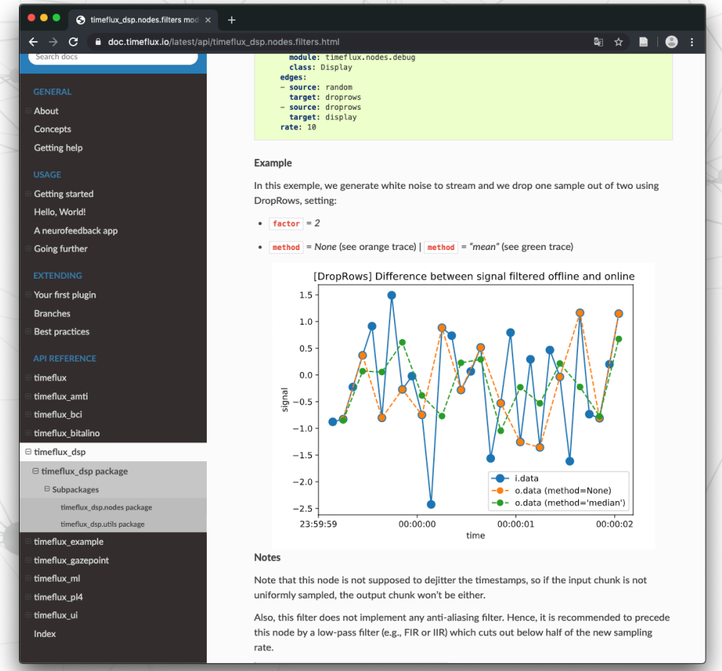
- **Networking**: Pub/Sub, Lab Streaming Layer, OSC, WebSocket
- **Recording and replay**: HDF5 file format
- **Digital Signal Processing**
- **Machine Learning**
- **User interface**: monitoring, web apps

Batteries included

- Multidimensional **matrix manipulation**: queries, transformations, expressions, epoching, windowing
- Native **device drivers**
- Sub-millisecond **synchronization**
- **Debugging** tools
- Pre and post **hooks**

Documentation

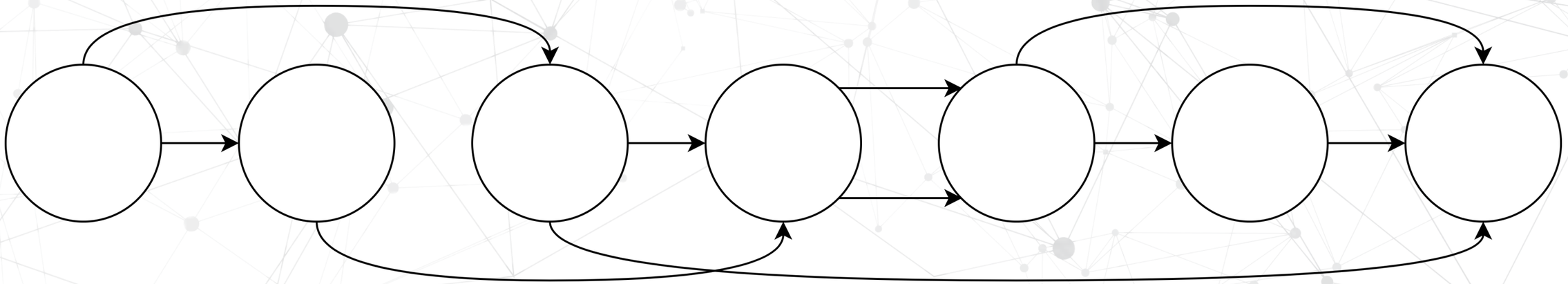




Pipelines

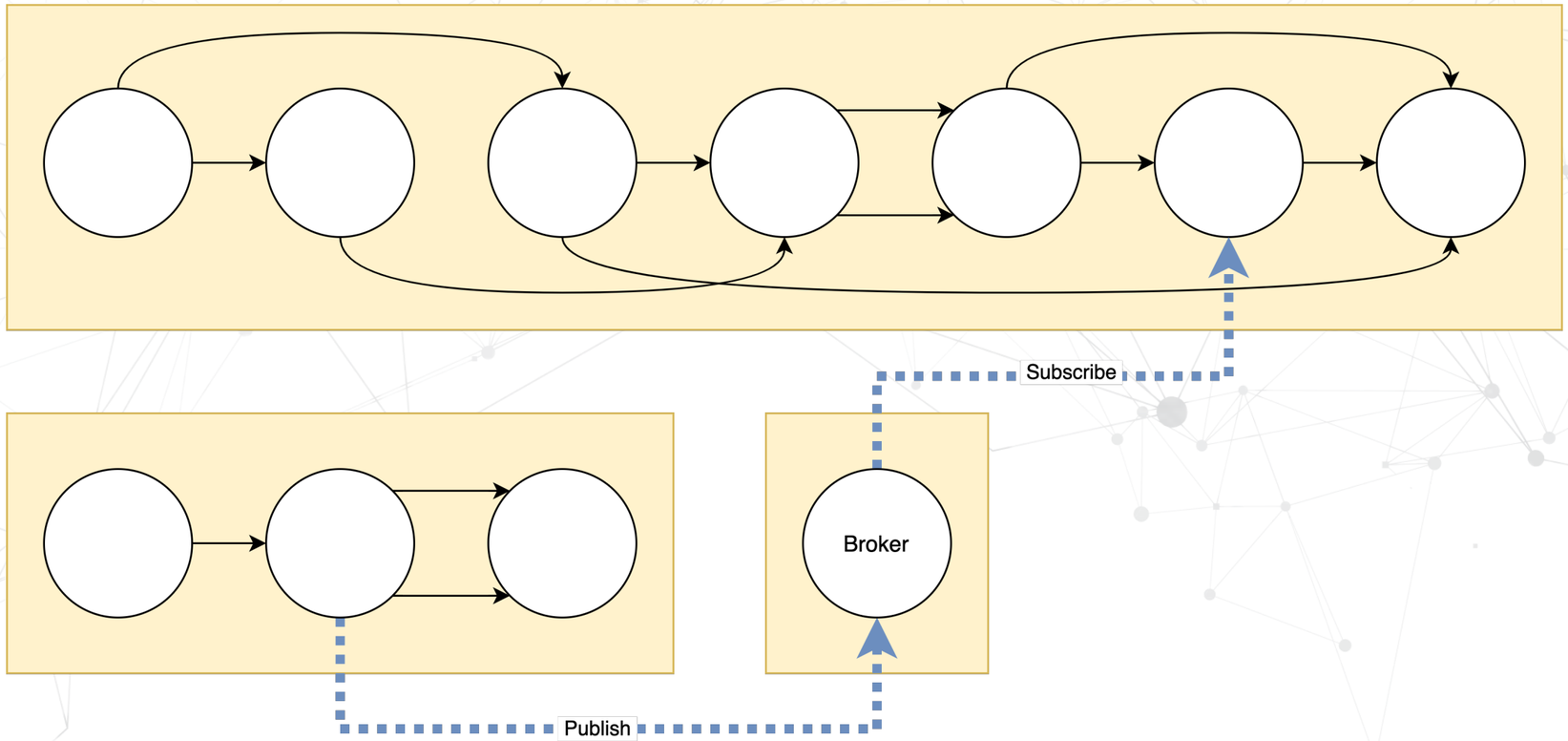
Directed Acyclic Graph (DAG)

A set of **nodes** connected by **edges**, where information **flows** in a given direction, **without any loop**.



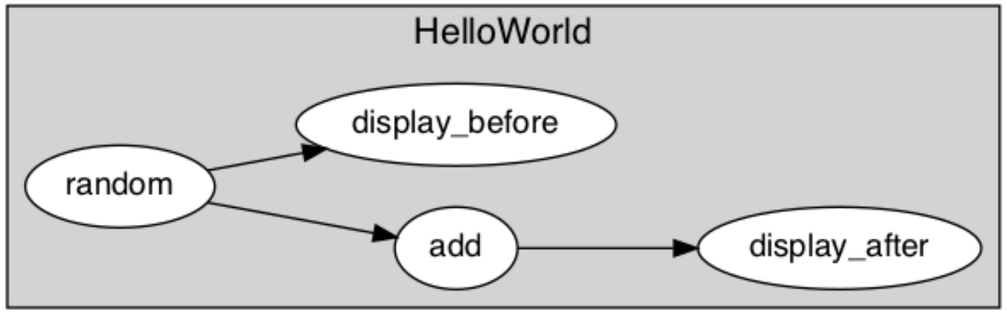
Multiple DAGs

Pub/Sub allows asynchronous loops without breaking anything.



helloworld.yaml

```yaml
graphs:
  - id: HelloWorld
    nodes:
    - id: random
      module: timeflux.nodes.random
      class: Random
    - id: add
      module: timeflux_example.nodes.arithmetic
      class: Add
      params:
        value: 1
    - id: display_before
      module: timeflux.nodes.debug
      class: Display
    - id: display_after
      module: timeflux.nodes.debug
      class: Display
    edges:
    - source: random
      target: add
    - source: random
      target: display_before
    - source: add
      target: display_after
    rate: 1
```



Run it!

```
timeflux -d helloworld.yaml
```

Confused?

https://doc.timeflux.io

Interfaces

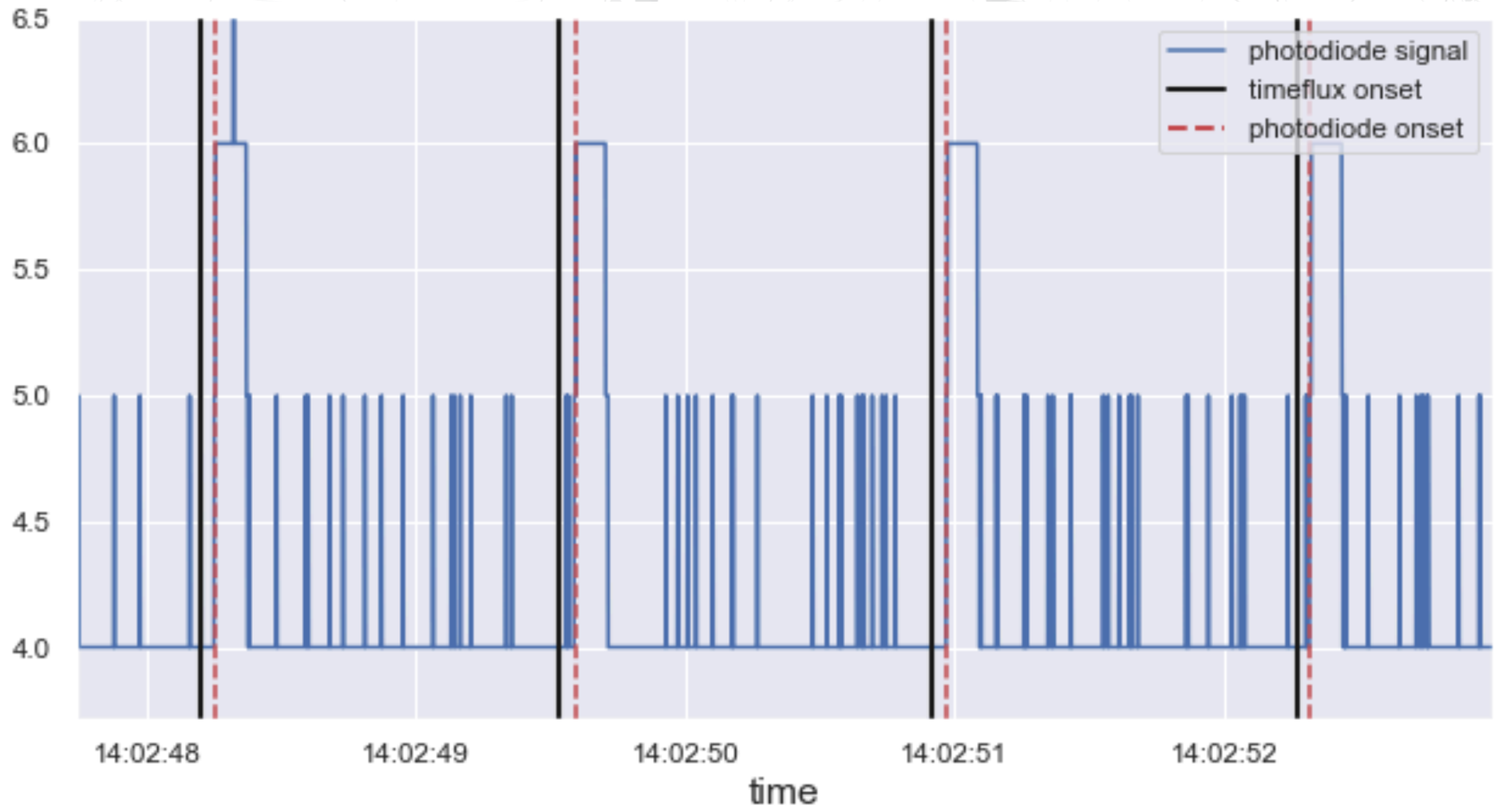
Timeflux.js

- A **JavaScript** API for:
  - Building **user interfaces** available from a browser
  - Receiving and sending data **streams** and **events**
  - Delivering **precisely scheduled** stimuli: suitable for SSVEP and ERP research

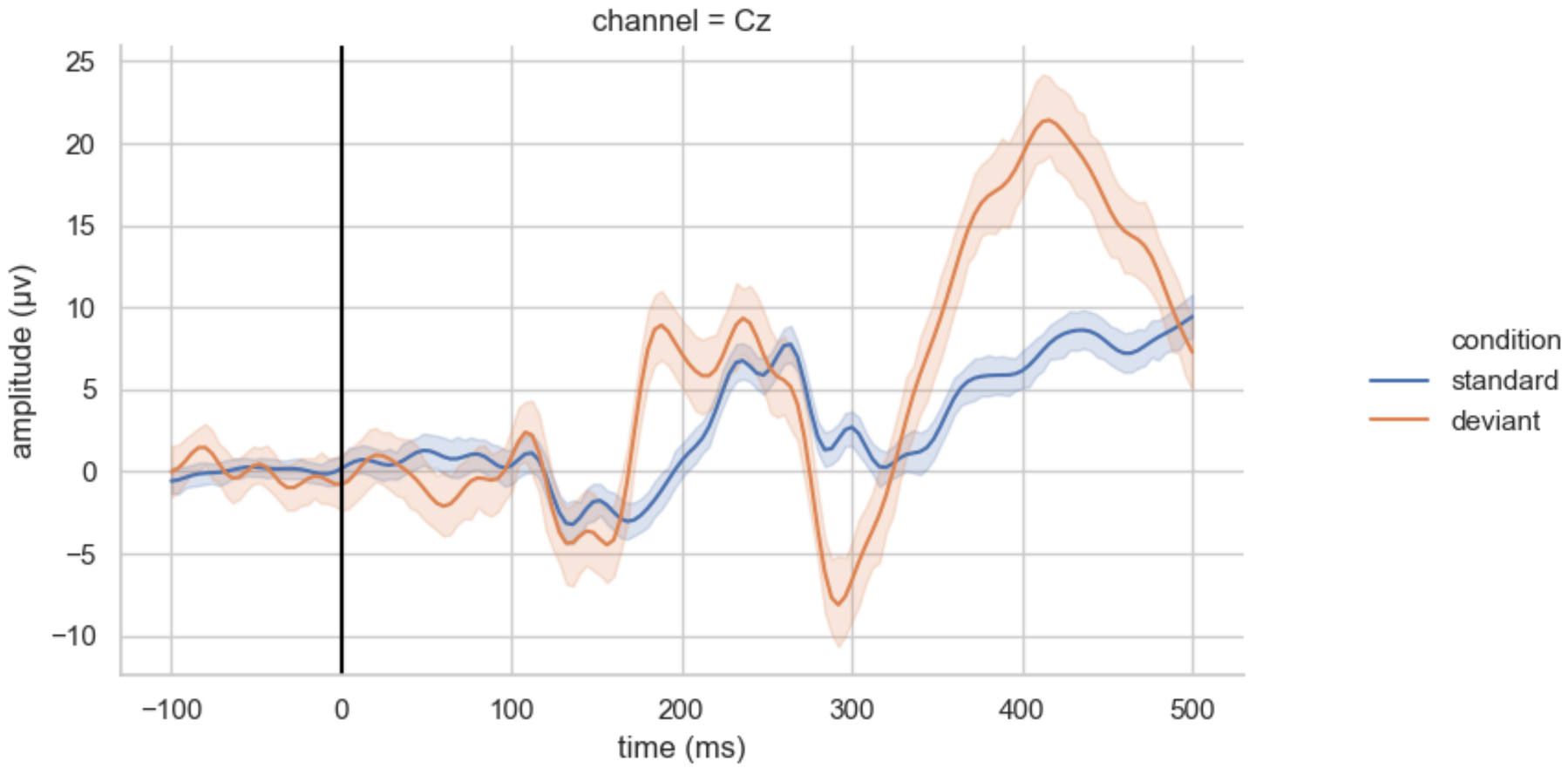
Stimulus presentation

- **Perfect timing** in a browser is **hard**
- But Timeflux.js makes it **easy**!
- Schedule **repeating** stimuli or **one-time** tasks
- Know **exactly** when the stimulus has been displayed
- Well tested in **Chromium**. Other browsers *may* have issues

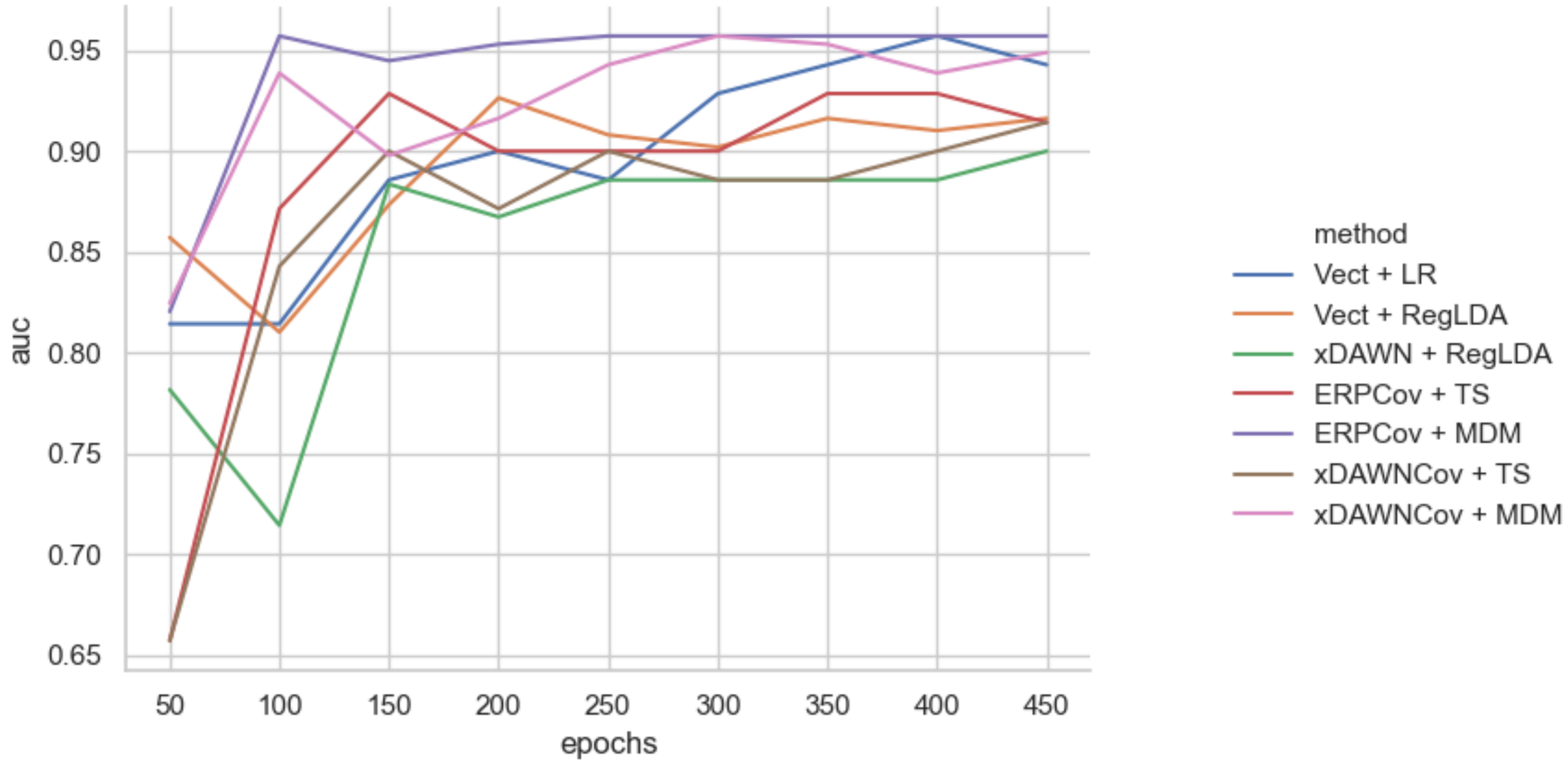
Validation: software event VS photodiode



Validation: evoked potentials

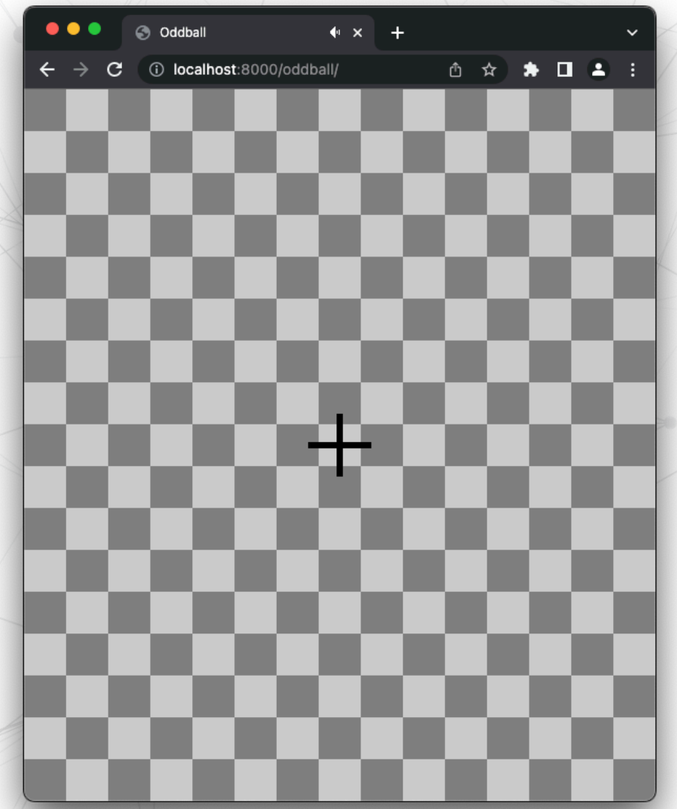


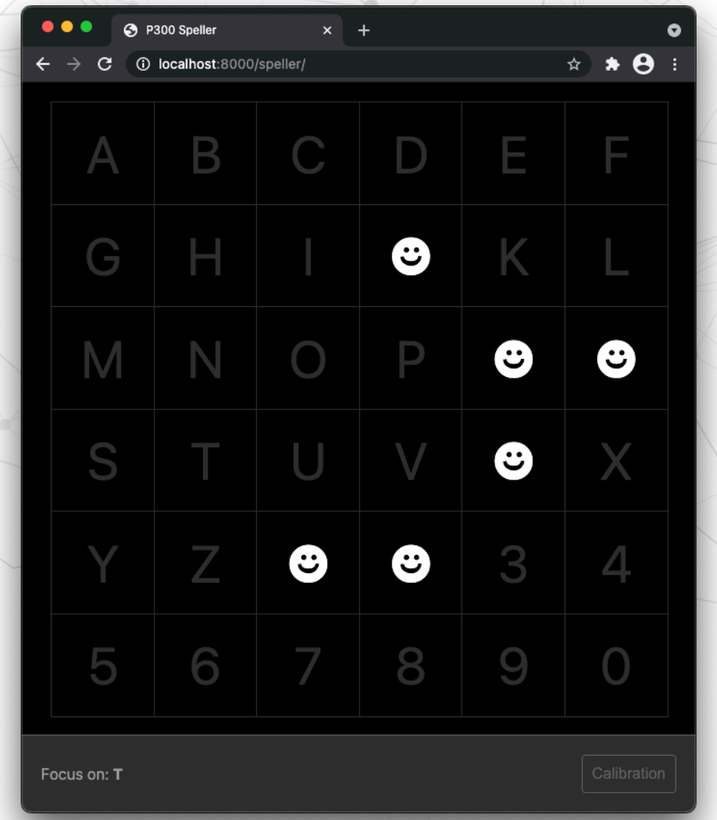
Validation: single-trial ERP classification



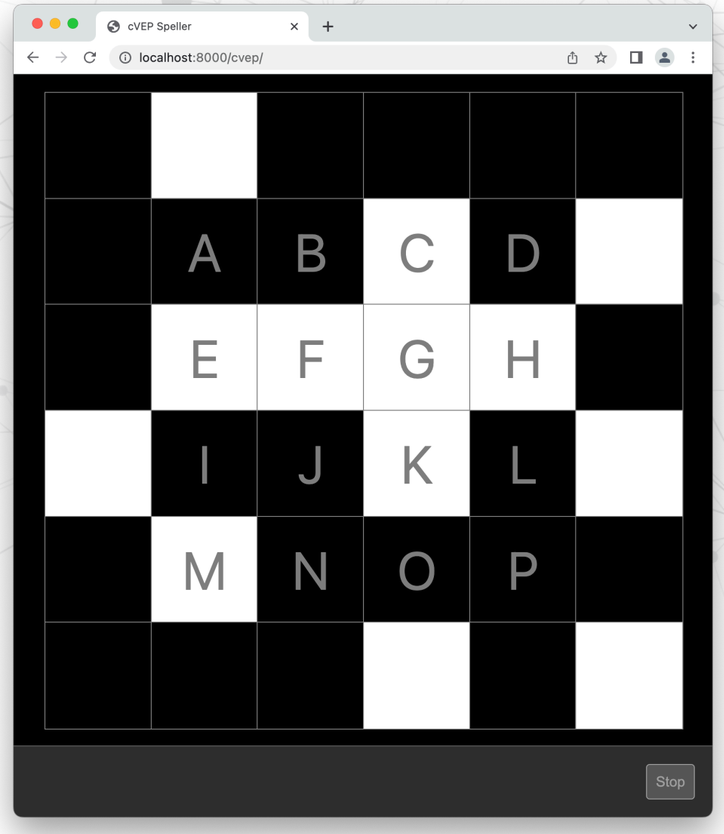
A few applications and plugins

P300 tools

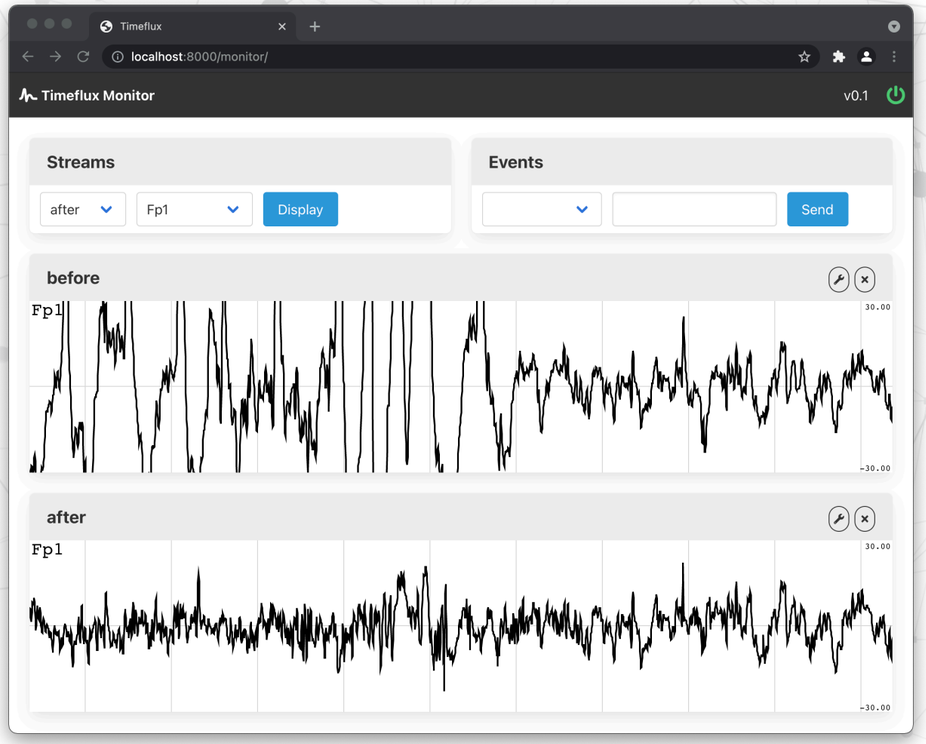

Oddball
localhost:8000/oddball/


P300 Speller
localhost:8000/speller/
A B C D E F
G H I K L
M N O P
S T U V X
Y Z 3 4
5 6 7 8 9 0
Focus on: T
Calibration

CVEP speller



Online denoising based on rASR [1]



[1] Blum, Sarah, et al. “A Riemannian Modification of Artifact Subspace Reconstruction for EEG Artifact Handling.”

And more...

- Standard **BCI paradigms** (EEG):
  - SSVEP
  - CVEP ✔
  - P300 ✔
  - Motor Imagery
- **Hyperscanning** (EEG) ✔
- **Neurofeedback** (EEG) ✔
- **Cardiac coherence** (ECG, PPG) ✔
- **Gesture detection** (EMG) ✔

Demos and code

https://github.com/timeflux/demos

Let's build a BCI speller!

Code modulation VEP [2]

23	24	25	26	27	28	29	30	31	0
31	0	1	2	3	4	5	6	7	8
7	8	9	10	11	12	13	14	15	16
15	16	17	18	19	20	21	22	23	24
23	24	25	26	27	28	29	30	31	0
31	0	1	2	3	4	5	6	7	8

T0 : 100000111000...1001111110

T1 : 101000001110...1010011111

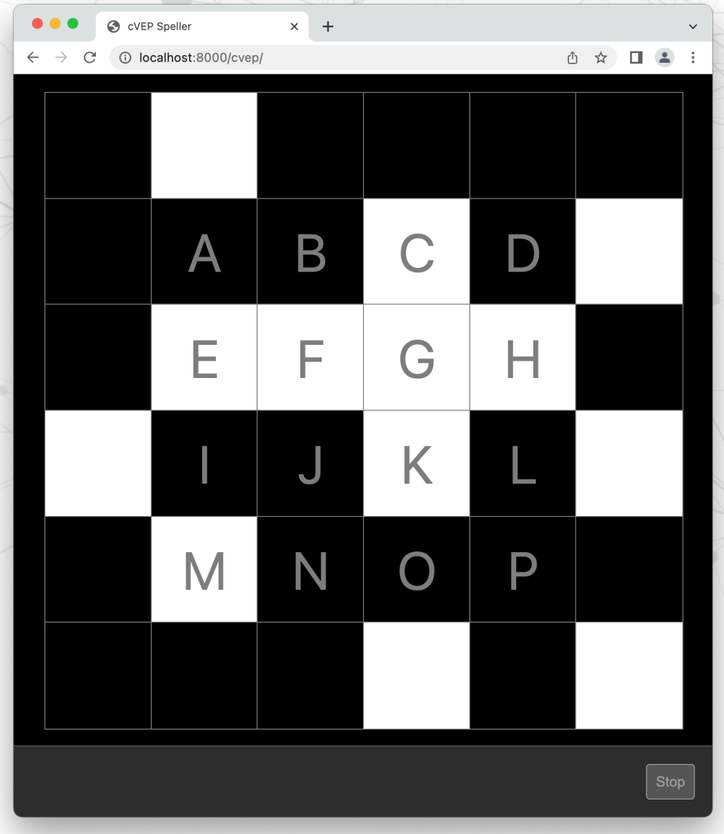
T2 : 111010000011...1010100111

T3 : 111110100000...0010101001

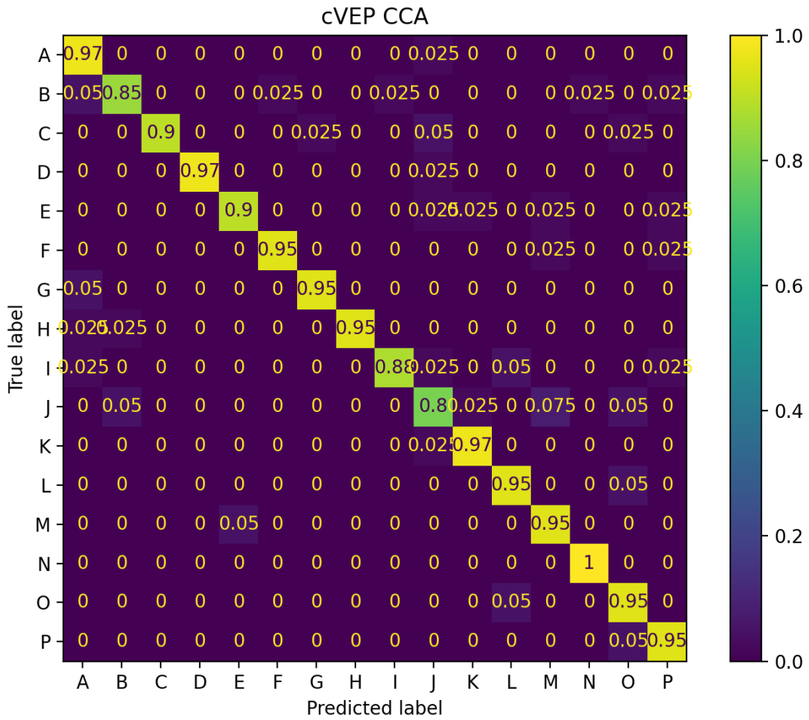
T31: 000001110000...0011111101

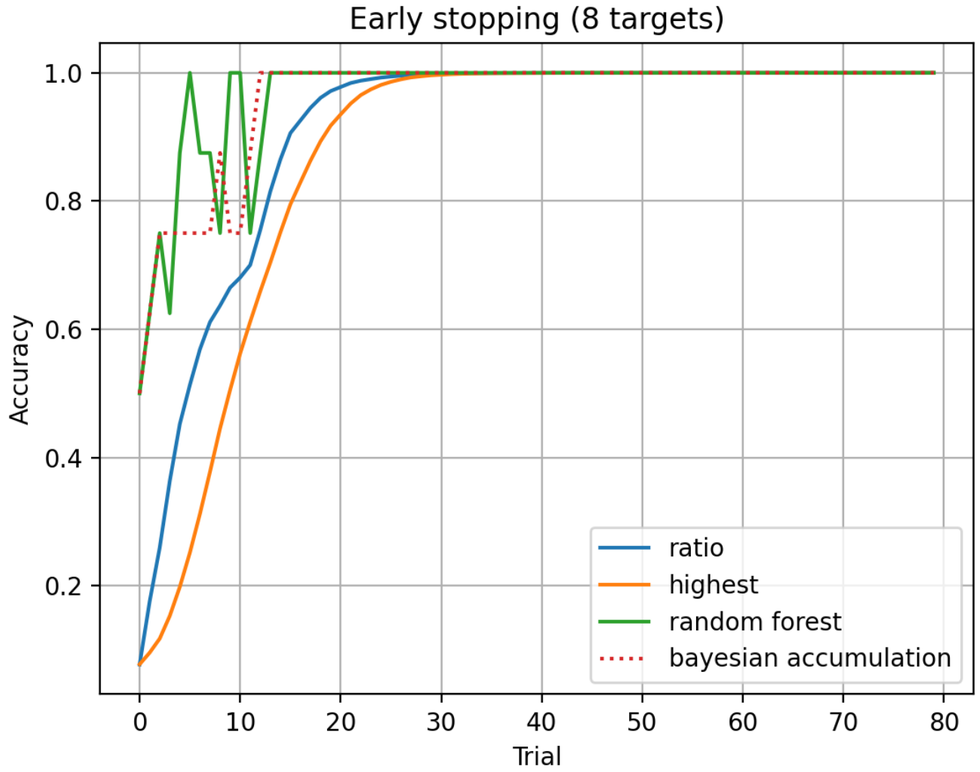
[2] Bin, Guangyu, et al. "A high-speed BCI based on code modulation VEP."

Let's dive in!



It works quite well :)





Bayesian accumulation and early-stopping [3]

BRAIN-COMPUTER INTERFACES
https://doi.org/10.1080/2326263X.2022.2140467

Taylor & Francis
Taylor & Francis Group

Check for updates

End-to-end P300 BCI using Bayesian accumulation of Riemannian probabilities

Quentin Barthélemy [a], Sylvain Chevallier [b], Raphaëlle Bertrand-Lalo [c] and Pierre Clisson [d]

[a]Foxstream, Vaux-en-Velin, France; [b]Université Paris-Saclay, UVSQ, LISV, Vélizy-Villacoublay, France; [c]Independent scientist, Nantes, France; [d]Independent scientist, Paris, France

ABSTRACT

In brain–computer interfaces (BCI), most of the approaches based on event-related potential (ERP) focus on the detection of P300, aiming for single trial classification for a speller task, but leaving the character classification out of the scope. To reduce the number of repetitions while maintaining a good character classification, it is critical to embrace the full classification problem. We introduce an end-to-end pipeline, starting from feature extraction, and composed of an ERP-level classification using probabilistic Riemannian MDM which feeds a character-level classification using Bayesian accumulation of confidence across trials. Whereas existing approaches only increase the confidence of a character when it is flashed, our new pipeline, called Bayesian accumulation of Riemannian probabilities (ASAP), update the confidence of each character after each flash. We demonstrate that this seamless processing of information from signal to BCI characters performs significantly better than standard methods on public P300 datasets.

ARTICLE HISTORY
Received 15 March 2022
Accepted 21 October 2022

KEYWORDS
Brain-computer interfaces; P300 classification; Riemannian geometry; character classification; Bayesian accumulation

1. Introduction

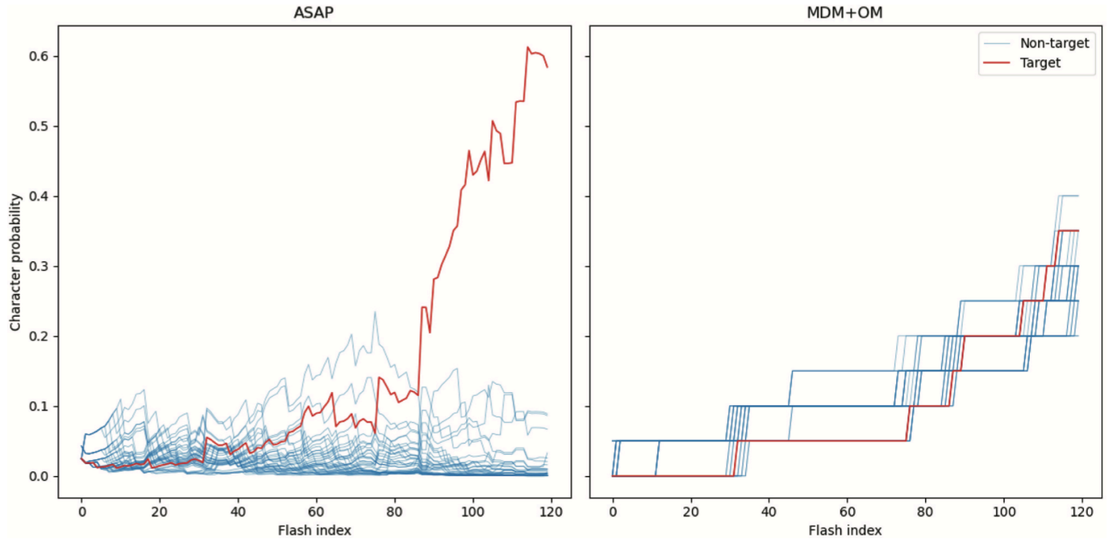
Brain-computer interfaces (BCI) have become a rising domain in neurotechnologies, allowing decoding brain activity, with multiple applications [1] using electroencephalography (EEG). Out-of-the-lab applications in EEG-based BCI face a difficult challenge: on top of the hard problem of decoding brain signals in real-time, most of the existing literature provides results that are difficult or even impossible to reproduce with online applications. A large portion of the BCI literature tackles processing or classification problem relying on offline analysis and dataset-specific preprocessing, or uses machine learning methods not suitable for real-world applications (long calibration process and model training phase). Indeed, the BCI community is continuously pushing toward increased reproducibility, robustness and acceptability of BCI, but the above-mentioned pitfalls are well documented [2–5].

In this study, we focus on a well-known class of BCI, that is based on the detection of event-related potentials (ERP) induced by an oddball effect. Common applications are P300-spellers [6–8] or P300-games, such as Brain Invaders [8] or Raccoons vs Demons [9]. P300 is an ERP elicited when the BCI flashes the target character (also called symbol or item). The BCI's task in these ERP-based paradigms is to detect the presence or absence of a P300 to retrieve the flashed character [6].

The low signal-to-noise ratio (SNR) of P300 waves in EEG signals makes it difficult to classify them on a single trial. Therefore, the first attempts to detect P300 waves consisted in averaging multiple trials (about 15 repetitions for each flashing character). While these attempts were able to successfully discriminate between target and non-target flashing [6], the resulting BCI was slow and the information transfer rate (ITR) [10] was low. To reduce the number of repetitions, further attempts applied advanced spatial filters to increase the amplitude related to P300 waves, like xDAWN which maximizes the signal to signal+noise ratio [7]. To further reduce the required number of repetitions, ERP detection was achieved with classifiers such as LDA or SVM trained on signals enhanced with spatial filters [11].

However, spatial filters are subject- and session-dependent [12–14], preventing cross-session and cross-subject transfer capabilities [8]. This implies that a subject should perform a careful calibration phase before being able to use the BCI; this calibration is required for each session even for someone using a BCI on a regular basis. Recently, Riemannian approaches applied to BCI have allowed the design of new classifiers, such as the minimum-distance-to-mean (MDM) [8], having high performances with shorter calibrations [4,8]. Whereas optimized spatial filters estimate enhanced signals, Riemannian BCI work in the space of covariance matrices to process and classify the

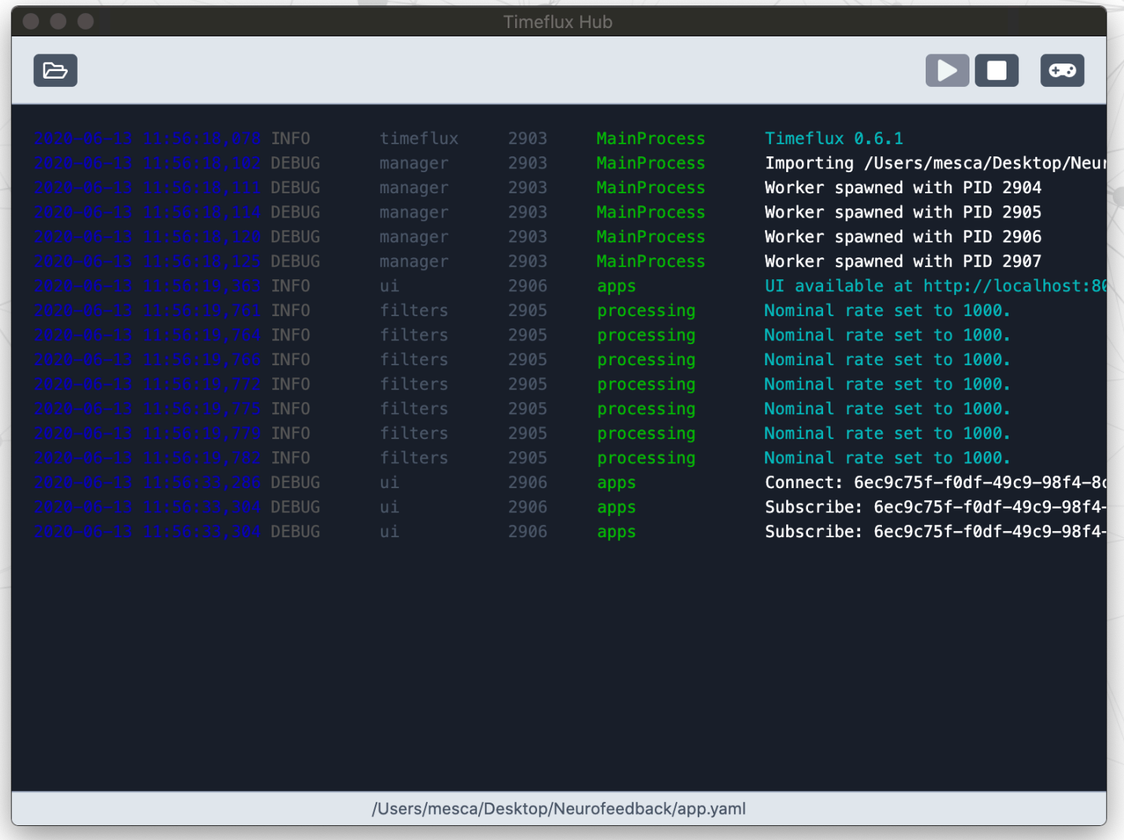
CONTACT Quentin Barthélemy q.barthelemy@gmail.com Foxstream, 6 rue du Dauphiné, 69120 Vaux-en-Velin, France



[3] Quentin Barthélemy, Sylvain Chevallier, Raphaëlle Bertrand-Lalo & Pierre Clisson (2023) "End-to-end P300 BCI using Bayesian accumulation of Riemannian probabilities."

Coming soon

Timeflux Hub



Conclusion

Getting help

- **Website**: https://timeflux.io
- **Documentation**: https://doc.timeflux.io
- **Bugs**: https://github.com/timeflux
- **Slack**: https://timeflux.io/slack
- **Email**: contact@timeflux.io

Sustainable open-source

Featured sponsor

Thanks!