 DOI: 10.3217/978-3-99161-050-2-086

# Redwood: A User Friendly Extension to the BCPy2000 System for Developing Advanced BCI Applications

**Z.V. Freudenburg*[1], E.J. Aarnoutse[1], E. Erdal[1], E.C. Offenberg[1], J. Berezutskaya[1], M.J. Vansteensel[1], N.F. Ramsey[1,2]**

[1]*Department of Neurology and Neurosurgery, University Medical Center Utrecht Brain Center, Utrecht University, Utrecht, The Netherlands;* [2]*Donders Institute for Brain, Cognition and Behaviour, Radboud University, Nijmegen, The Netherlands.* ·P.O. Box 1234, Utrecht, Netherlands. E-mail: Z.V.Freudenburg@umcutrecht.nl

*Introduction:* BCI2000 [1] is a widely used software platform in BCI research. It has been integrated with a large range of brain signal acquisition modalities and hardware systems and has been used in many BCI research paradigms. The growing trend towards using advanced machine learning tools for online brain signal analysis has also led to the integration of Python, which provides cutting edge machine learning tools, into BCPy2000 [2]. While this allows for both Python-based signal processing (SP) and application (App) development it runs the Python code within the traditional C++-based BCI2000 framework. A fundamental feature of this framework is the process block cycle that ties the speed at which BCI Apps can update the feedback screen to the computational time needed to acquire, process and save new brain data. While BCI2000 has proven itself in the BCI research arena, we present Redwood as an extension to the BCPy2000 framework that aids in advanced BCI App development.

*Materials and Methods:* **BCI2000** is a modular system with 3 modules for: signal acquisition, signal processing (SP), and BCI application logic (App). These 3 modules are governed by an Operator module that calls each module in sequence each process cycle. The Operator process cycle is determined by the 'block size', which is the number of data samples acquired, processed and responded to by the App. Communication between the modules is achieved via 'state' variables that can be defined, read, and written by each module and are saved to a 'dat' file with the block of raw data samples each process cycle. **BCPy2000** supports Python use in the SP and App modules. The **Redwood** system builds on the BCPy2000 framework by adding a thread (Screen) that runs a PyQt based application parallel to App and accesses state variables via the BCI2000Remote class. In this way the BCPy2000 App logic that updates state values and responds to SP output is split from logic updating the visual feedback to the user. Redwood also opens a RedwoodOperator window to provide direct interaction with App.

*Results:* Building on BCPy2000, Redwood inherits its advantages while adding features that ease Python application development. By decoupling the Screen logic from that of App the update rate of the user feedback screen is no longer tied to the BCI2000 process cycle. This allows for two advantages; Firstly, complex logic defining the visual response to BCI events (e.g. computing AI agent behaviour or using language models for word completion) can be spread across multiple App cycle blocks. Secondly, the refresh rate of the Screen can be faster than the App cycle block. Given that with a BCI system of 100+ data channels and advanced SP a BCI2000 cycle time below 100ms is challenging, when a user initiates a screen action it will be made at 10fps if the screen is updated by App. However, Screen can easily meet the lower bound of 30 fps often needed for acceptable gaming interaction. Additionally, the RedwoodOperator allows the App to be started after the Screen window has been resized and positioned for the user, to be paused without stopping the .dat file recording, and to give real-time feedback about states. Finally, Screen can also create a log of screen events that happen in response to App state changes at the screen fps time scale.

*Conclusion:* Redwood expands capabilities for App development with Python while preserving all the benefits the BCI2000 platform offers.

*Acknowledgments and Disclosures:* The research is funded by NIDCD U01DC016686, NINDS UH3NS114439, and EU EIC-101070939. No disclosures to report.

*References:*

[1] Schalk G, McFarland DJ, Hinterberger, T, Birbaumer, N, Wolpaw JR, *BCI2000: a general-purpose brain-computer interface (BCI) system.*, IEEE Transactions on biomedical engineering, 51(6), 1034-1043, 2004.

[2] Hill NJ, Schreiner T, Puzicha C, Farquhar J, *BCPy2000*, http://bci2000.org/downloads, 2007.



 Published by Verlag der Technischen Universität Graz