Context-Aware Python Middleware for BCI-Controlled Desktop Automation

Ravikiranreddy Karnati, Chakradhar Kadali

(Department of Computer Science, Hyderabad Institute of Technology and Management,, hyderabad, india
Email: ravikiranreddykarnati@gmail.com)

(Department of Computer Engineering, Hyderabad Institute of Technology and Management, india
Email: chakri262005@gmail.com)

Guide: Mr. T. Raghavendra Gupta, Associate Professor, Department of CSE

Abstract:

Brain-computer interfaces (BCIs) provide a potential means of hands-free human-computer interaction by translating brain-derived signals into commands. However, mapping a small command vocabulary to operations across multiple desktop applications creates software-engineering challenges involving command ambiguity, repeated activation, application context, and execution safety. This paper presents the design of a context-aware Python middleware framework intended to connect discrete BCI commands supplied by the Emotiv Cortex environment to desktop application automation handlers. The framework uses four command labels—PUSH, PULL, LEFT, and RIGHT—and combines command normalization, confidence-threshold filtering, temporal cooldown, domain-state management, command lookup, and inter-process communication. The documented design maps these commands to more than twenty operations across seven application domains: YouTube, Notepad, Microsoft Excel, Microsoft Word, Microsoft PowerPoint, email clients, and web browsers. A simulation-based benchmark reported in the project material used 1,000 simulated command-stream bursts. The reported mean middleware latency was 12.40 ms with a standard deviation of 2.10 ms; the project material also reports 95.8% duplicate-trigger suppression and 100% routing success for the tested cases. These figures describe software-level simulation results and must not be interpreted as EEG classification accuracy, human-subject performance, or validation with a physically connected Emotiv headset.

Keywords **--- Brain-computer interface, EEG, Emotiv Cortex API, context-aware middleware, desktop automation, command routing, debounce, simulation-based evaluation, human-computer interaction.**

I. INTRODUCTION

Brain-computer interfaces (BCIs) establish a communication pathway between brain activity and external devices. Non-invasive electroencephalography (EEG) is widely studied because it records electrical activity from the scalp without requiring an implanted interface. BCI research has explored communication, assistive interaction, motor-imagery control, and other applications [1], [3], [6].

A practical desktop environment presents a different challenge from selecting one command in a controlled BCI task. Productivity software, browsers, and media applications expose many operations, while a BCI interface may provide only a small set of discrete command labels. A direct one-to-one mapping between each mental command and a global operating-system action can therefore be restrictive. It can also make the same command ambiguous when the user changes applications.

This paper addresses that software orchestration problem through a context-aware Python middleware design. The middleware is intended to receive discrete command events from an Emotiv Cortex API connection, normalize them, apply confidence and timing filters, determine the active application domain, and dispatch an application-specific payload to a desktop client agent. The paper focuses on this middleware and its simulated software-level evaluation. It does not report a completed human-subject experiment or a live EEG headset validation.

The main contributions are: (i) a layered architecture separating BCI vendor communication from application automation; (ii) a stateful command-routing model that reuses four command labels across seven application domains; (iii) command filtering and cooldown logic intended to reduce repeated execution; (iv) safety checks for selected destructive spreadsheet operations; and (v) a preliminary report of simulated middleware timing and routing results.

II. PROBLEM STATEMENT AND RESEARCH QUESTIONS

The system addresses the gap between a limited set of discrete BCI command labels and the larger action vocabulary required by desktop applications. Three software-level risks motivate the middleware: command ambiguity when the active application changes; repeated execution when similar command events arrive within a short interval; and unintended execution of destructive operations such as worksheet deletion or data overwriting.

The work is guided by the following research questions:

RQ1: What middleware processing latency is reported by the software benchmark for command normalization, debounce checking, domain lookup, and local socket communication?

RQ2: How can active-domain state be used to map a small command vocabulary to different application-specific actions?

RQ3: Which software guardrails are represented in the command-routing design for selected destructive operations?

III. RELATED WORK AND RESEARCH GAP

A. BCI Communication and Control

Wolpaw et al. describe BCI systems for communication and control and establish the broader technical context for translating brain activity into device commands [1]. Pfurtscheller and Neuper discuss motor imagery and direct brain-computer communication [3]. These works provide background for BCI interaction but do not, by themselves, validate the middleware proposed here.

B. EEG Classification and BCI Systems

Lotte et al. review classification algorithms for EEG-based BCIs and discuss the algorithmic challenges associated with interpreting EEG signals [4]. Al-Saegh et al. review deep-learning approaches for motor-imagery EEG-based BCIs [5]. BCI2000 is described by Schalk et al. as a general-purpose BCI system [10]. These publications are relevant to the signal-processing and system-integration context. The present work does not propose a new EEG classifier and treats the command labels as outputs supplied to the middleware.

C. Assistive Technology and Practical BCI Deployment

Millán et al. discuss the combination of BCIs and assistive technologies, including the practical challenges of translating BCI outputs into useful assistance [6]. McMullen et al. demonstrate a semi-autonomous hybrid BCI in a robotic-arm context [8]. Allison et al. discuss biological, technological, and social issues involved in practical BCI development [9]. These works support the need to consider the complete interaction stack.

IV. SYSTEM DESIGN AND ARCHITECTURE

The middleware design is modular. The Cortex connector encapsulates WebSocket communication; the command normalizer validates incoming command labels and confidence values; the debounce engine suppresses events arriving within the configured cooldown; the domain manager stores the active application context; and the dashboard IPC server exchanges execution payloads and acknowledgements with the client agent.

The architecture is intended to isolate application automation from the vendor-specific BCI interface. This separation allows the routing and client-handling logic to be developed and tested using simulated command events even when a physical headset is not connected.

V. SIMULATION-BASED EVALUATION METHODOLOGY

The evaluation reported in the project document is a software-level simulation, not a live BCI experiment. The stated host environment was Windows 11 with an Intel Core i7-12700H processor and 16 GB RAM. The project document reports 1,000 simulated command-stream bursts and timing with Python's high-resolution performance counter.

The reported latency stages are JSON-RPC parsing and normalization, debounce and cooldown checking, domain-state lookup and mapping, and local socket IPC transmission. The project document reports a mean total middleware latency of 12.40 ms (SD = 2.10 ms). It also reports 95.8% duplicate-trigger suppression with a configured cooldown of 0.8 seconds and 100% routing success in the tested simulation cases.

VI. RESULTS AND DISCUSSION

The reported middleware timing suggests that the software orchestration layer can process simulated command events with a low measured overhead on the stated host configuration. The largest component in the supplied breakdown is local socket IPC transmission (8.79 ms mean), followed by JSON-RPC parsing and normalization (1.82 ms), domain lookup and mapping (1.14 ms), and debounce checking (0.65 ms).

The domain-aware routing model provides a way to reuse a limited command vocabulary across several application contexts. This is a software design benefit; the study does not establish that four commands are easy to learn, cognitively efficient, or reliable for users with motor impairments. Those claims require a human-subject evaluation.

The safety logic illustrates why application context and action risk should be considered before dispatch. The protected-sheet check is a useful example, but destructive operations should ideally require explicit confirmation, maintain recoverable state, and produce auditable logs.

VII. LIMITATIONS AND THREATS TO VALIDITY

1. No live Emotiv EPOC headset acquisition or Cortex API session was performed for the reported benchmark.

2. No human participants were recruited; usability, workload, learning time, and accessibility outcomes were not measured.

3. The benchmark used simulated command bursts, but the generator, event distribution, raw logs, and exact test-case counts were not supplied with the manuscript.

4. The definitions and denominators for the reported 95.8% duplicate suppression and 100% routing success figures require documentation.

5. The 0.50 confidence threshold and 0.8-second cooldown are documented configuration values; no parameter-sensitivity analysis is available.

VIII. CONCLUSION AND FUTURE WORK

This paper presented a context-aware Python middleware design for mapping a small set of discrete BCI command labels to operations across multiple desktop application domains. The design combines command normalization, confidence filtering, temporal cooldown, active-domain state management, application-specific lookup, local IPC, and selected execution guardrails. The project material documents mappings for more than twenty actions across seven application domains.

A simulation-based benchmark reported a mean middleware latency of 12.40 ms (SD = 2.10 ms) across 1,000 simulated command-stream bursts. The project material also reports 95.8% duplicate-trigger suppression and 100% routing success for the tested cases, although the underlying metric definitions and raw data must be supplied to make these percentages reproducible.

Future work should first publish the simulation harness, test vectors, raw timing logs, and precise metric definitions. The next stage should validate the Cortex API connection with a physical Emotiv headset, quantify command recognition and end-to-end latency, test failure and recovery scenarios, and conduct an appropriately approved user study.

ACKNOWLEDGEMENT

The authors thank their internal supervisor and guide, Mr. T. Raghavendra Gupta, Associate Professor, Department of CSE, for technical guidance and constant encouragement, and the Program Head and the Hyderabad Institute of Technology and Management for the facilities provided.

REFERENCES

[1] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan, "Brain-computer interfaces for communication and control," Clinical Neurophysiology, vol. 113, no. 6, pp. 767-791, 2002.

[2] Emotiv Inc., "Cortex API v2 documentation and developer guide," Emotiv Knowledge Base, 2024. [Online]. Available: https://emotiv.gitbook.io/cortex-api/

[3] G. Pfurtscheller and C. Neuper, "Motor imagery and direct brain-computer communication," Proceedings of the IEEE, vol. 89, no. 7, pp. 1123-1134, 2001.

[4] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy, and F. Yger, "A review of classification algorithms for EEG-based brain-computer interfaces (2007-2017)," Journal of Neural Engineering, vol. 15, no. 3, p. 031005, 2018.

[5] A. Al-Saegh, S. A. Dawwd, and J. M. Abdul-Jabbar, "Deep learning for motor imagery EEG-based brain-computer interfaces: A review," Biomedical Signal Processing and Control, vol. 63, p. 102172, 2021.

[6] J. del R. Millán, R. Rupp, G. R. Müller-Putz, R. Murray-Smith, C. Giugliemma, M. Tangermann, and D. Mattia, "Combining brain-computer interfaces and assistive technologies: state-of-the-art and challenges," Frontiers in Neuroscience, vol. 4, p. 161, 2010.

[7] B. Blankertz, K.-R. Müller, D. J. Krusienski, G. Schalk, J. R. Wolpaw, A. Schlögl, and G. Pfurtscheller, "The BCI competition III: Validated data sets for BCI research," IEEE Transactions on Biomedical Engineering, vol. 53, no. 8, pp. 1615-1628, 2006.

[8] D. P. McMullen, G. Hotson, K. D. Katyal, B. A. Wester, M. S. Fifer, T. G. McGee, and N. E. Crone, "Demonstration of a semi-autonomous hybrid brain-computer interface using a sensory-guided robotic arm," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 22, no. 4, pp. 784-796, 2014.

[9] B. Z. Allison, S. Dunne, T. Lüth, and J. R. Millán, Towards Practical Brain-Computer Interfaces: Biological, Technological, and Social Issues. Springer, 2012.

[10] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw, "BCI2000: a general-purpose brain-computer interface (BCI) system," IEEE Transactions on Biomedical Engineering, vol. 51, no. 6, pp. 1034-1043, 2004.