The Unlock Software: A Modular Human-Machine Interface Software Framework

Byron V. Galbraith, Graham E. Voysey, Frank H. Guenther

1. Introduction

Researchers have spent considerable effort over the years developing methods that enable motor-impaired people to interact with and communicate through computerized systems. These methods exploit whatever reliable signals the user is able to generate, such as eye gaze fixation or bioelectric activity modulation, in order to drive the behavior of specialized applications.

Typically, a researcher will determine which signal source to base a control system on, devise a decoder that translates acquired signals into computerized actions, then design an experiment to validate and refine the decoder. A major challenge for researchers in this process is the actual development and execution of the software needed to facilitate the experiments, especially in the case of closed-loop scenarios where real-time processing and feedback to the user is required.

Tools have been developed to aid in this task. In the case of brain-computer interfaces (BCI), which drive the computerized system through detected changes in brain activity, BCI2000 [ref] is a popular choice. Other tools, such as PsychToolbox for MATLAB and PsychoPy for Python, provide a general set of basic methods for developing stimulus-driven experiments. The latter tools simplify the process of creating experimental software, but still require the researchers to write significant amounts of code and integrate with hardware or network data sources directly. The domain-specific packages may provide out-of-the-box interfaces for addressing these problems, but constrain what kind of experiments can be designed to what the developers of the packages provide. Going outside of this requires writing custom code, often in C++. This may lie outside the realm of a research lab's core competencies resulting in either the need to hire or collaborate with a research engineer skilled in software development, a significant productivity hit while researchers learn how to work with the code, or precluding the experiment all together.

To help solve this problem, we have developed The Unlock framework. Unlock is a software system written in the Python programming language that supports the creation of bio-signal interface applications (available: https://github.com/NeuralProsthesisLab/unlock). These applications can be either research-driven experiments or purely user-centric apps, such as a game or communication interface.

The primary goal of Unlock is to provide a free, open source platform that can separate the user interface and interaction logic of an application from the data acquisition and decoding underpinnings in order to allow both motivated developers and researchers the ability to contribute. It accomplishes this by decoupling and compositing various commonly used components together in order to create the applications. For instance, data acquisition and signal processing components, such as task-based decoders, can be designed by experts while adhering to a common application programming interface (API). This API ensures that a developer creating an entertainment app would not necessarily need to know which particular paradigm ends up being used as long as their application responded to specific defined event notifications and messages.

2. Materials and Methods

We designed Unlock to appeal to a broad array of researchers, who may be interested in acquiring many types of signal such as electroencephalograms (EEG), electromyograms (EMG), or eye gaze position; via many bio-signal probing modalities. The base system therefore includes default functionality for recording, decoding, and acting on a broad array of relevant signals. The primary focus is on visual BCI via EMG, but EMG and eye tracking recording are also supported. Since experimental configurations may differ widely, Unlock has been written in a way that makes it easily extensible and

customizable by individuals, while allowing them to focus on their application instead of the intricacies of the underlying software architecture.

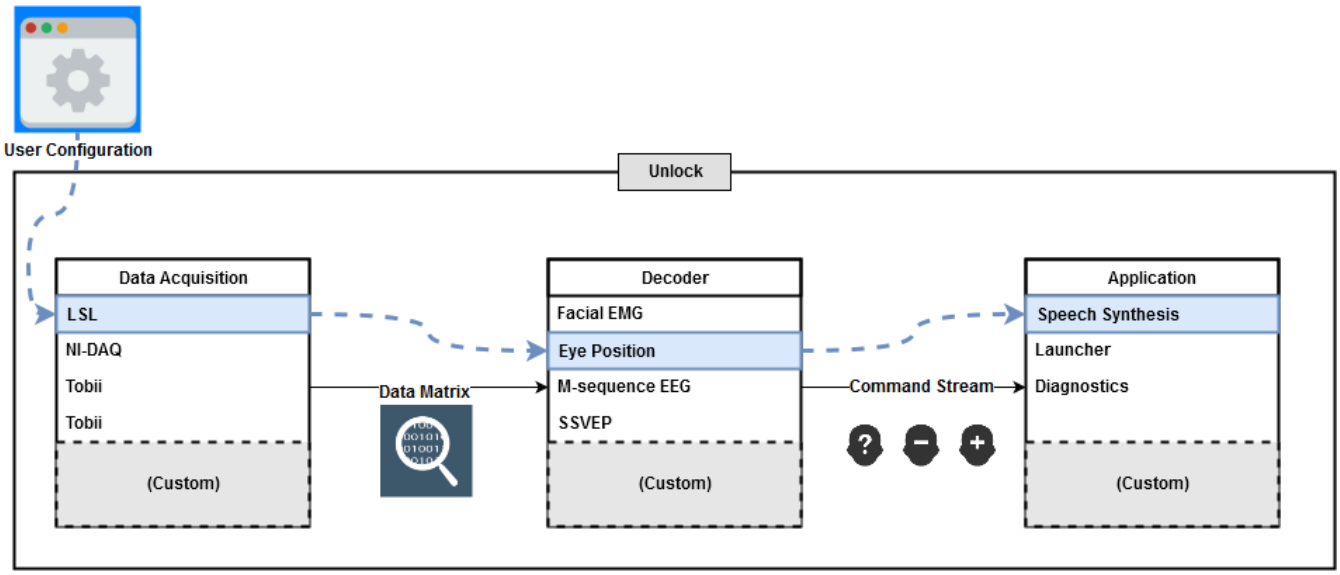

FIGURE 1. FLOW CONTROL

The conceptual flow of the stages of the Unlock workflow. In the Data Acquisition stage, biophysical measurements such as EEG or eye position are acquired from sensors and stored in memory as a stream of raw data. In the Decoder stage, these data are filtered and processed based on knowledge of what they represent, and broadcast as a sequence of decisions such as a directional command. In the Application stage, arbitrary user interations are driven by one or more command streams from one or more decoder. An example data flow based on a stored configuration is highlighted in blue which represents the use of Unlock to drive a speech synthesis application based on eye tracking, using the Lab Streaming Layer as the data recording interface.

Unlock incorporates two related concepts to describe measured biological data and use it to control the application: Commands and Decisions. Commands map measured signals to motion on the screen -- e.g., movement of a cursor by a certain number of pixels over a certain time period. Decisions map measured signals to actions -- e.g., a selection of an icon on a screen.

The components of a full BCI have been broken down into a series of discretized plugin modules. Primarily, these include Drivers, Decoders, and Apps. A Driver is the component that acquires data from exactly one hardware device from an arbitrary number of biological recording sites (e.g., one multi-electrode cap is treated as one Driver source), and provides it to the rest of Unlock in an unmodified form. Examples of devices that would be connected to Unlock as a Driver include EEG systems, eye trackers, and general data acquisition hardware, or a standard keyboard or mouse for testing and development. A Decoder takes the result of a Driver, and through a series of internal steps of filtering, signal processing, etc., produces a stream of Commands and Decisions. Apps subscribe to one or more Decoder streams, and use the Commands and Decisions it they obtain to perform arbitrary interactive tasks.

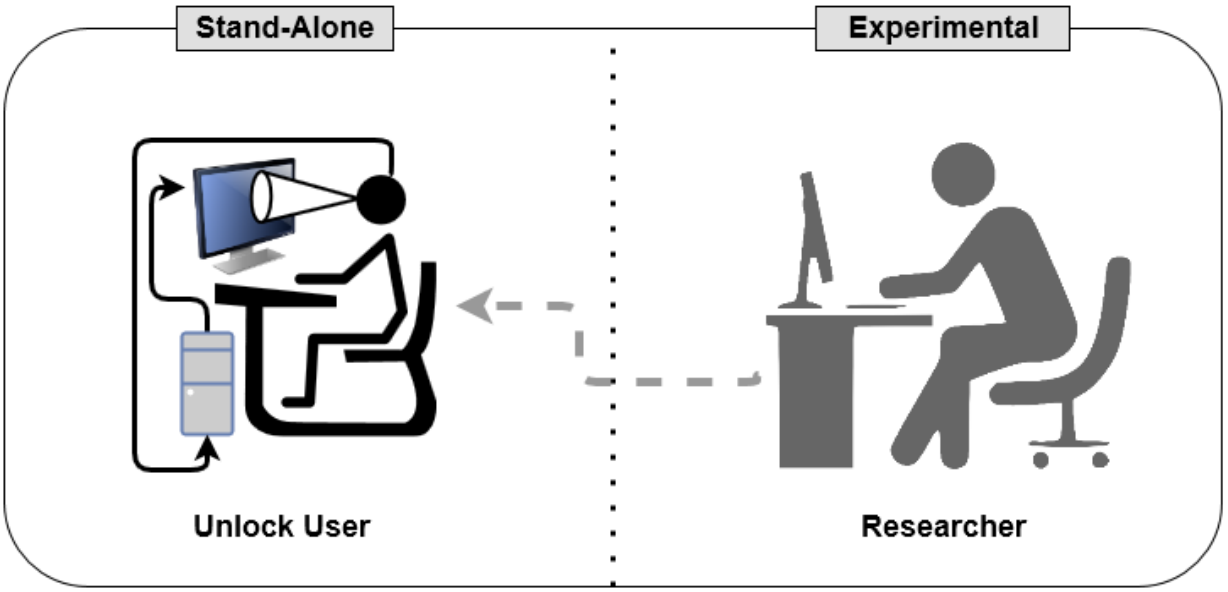

FIGURE 2. The Unlock Framework is suitable for use both as a stand-alone or experimental tool. In the stand-alone mode, we depict a user interacting with a computer system through visual control with no other input devices present; with the optional addition of a researcher, the Unlock system may be modified on-line, or data from user performance may be recorded remotely.

Apps are designed with the Model-View-Controller (MVC) pattern, a common way to separate concerns in a software application. Models subsume the "business logic" of the app, with no knowledge of or control over the display. Views encapsulate all graphical presentation of App behavior, and Controllers transfer changes in the Model state into updated View display parameters.

Several common Drivers and Decoders are provided in the base version of Unlock. These include Windows driver support for all National Instruments DAQ-MX series interface boards, the Tobii EyeX eye tracker, and keyboard/mouse input for development and testing. By default, several Apps are included; among which are diagnostic tools for systems testing, as well as a Dashboard, which allows selection and use of all other developed and installed applications.

Additionally, an App generation tool is included. In an interactive session, a functional basic app template can be generated by specifying which options are required, and the result is a small Python module whose functionality is fully pre-integrated into Unlock, but whose code is separate and easily maintained.

Unlock incorporates an interface to the Lab Streaming Layer (LSL) (cite). LSL is a data interface which standardizes the data formats produced by 20 of the most commonly used EEG recording hardware devices, as well as eye trackers, handheld controllers such as the Wiimote, and motion-capture hardware. Crucially, it is designed to provide time-series data across multiple devices simultaneously with precise timing, even with inconsistent connections between hardware and recording software, simultaneous multiple timescales, or data that is not easily timed. This positions Unlock to be very flexible about the data it can consume and support many hardware acquisition systems natively without requiring investigators to write complex data acquisition routines.

In the case where Unlock does require more complicated modification, we provide an easy framework by which new drivers and decoders can be developed for custom needs. The base version of Unlock provides an Application Programming Interface (API) which simplifies the integration of new Apps and Decoders and Drivers into the Unlock framework. Developers who seek to integrate a new App, for example, can focus on the experimental logic of their application rather than the complexities of graphics programming, all of which is abstracted into commonly used methods.

3. Results

The features built into Unlock provide researchers with the capabilities to support a wide array of hardware while being able to focus on experimental design and subject data acquisition.

A suite of default applications, drivers, and decoders have been developed for Unlock. Earlier versions of some applications were previously described in (Brumberg 2012).

Eight subjects (2 females, aged 21-38) were recruited to perform a series of Unlock experiments, with data collected over a single session. All gave informed consent for the study, which had been approved by the Boston University Institutional Review Board. Two subjects had prior BCI experience, though none had experience with c-VEP BCI. All subjects had normal or corrected to normal vision, and none reported a history of epilepsy or indicated sensitivity to rapidly flicking lights.

EEG was recorded using the Enobio 8 (Neuroelectrics, Barcelona, Spain), an eight-channel wireless EEG recording system. Electrodes were placed at locations PO7, O1, Oz, O2, PO8, PO3, Pz, and PO4 according to the 10-20 international system with reference CMS and DRL electrodes placed over the right mastoid. EEG was digitized at 500Hz and transmitted via a Bluetooth connection to the BCI computer, where it passed through the Neuroelectrics NIC software to the Unlock application and saved to disk.

The experimental sessions were conducted in in an office-like environment lit with fluorescent ceiling lights. Subjects were seated in a comfortable chair approximately 70cm from an LCD computer monitor with a display resolution of 1920x1080 pixels operating at a 120Hz refresh rate. A Tobii EyeX eye tracker mounted just under the screen was used to track the subjects' gaze during trials to ensure proper gaze fixation.

Prior to beginning the experiment, subjects created eye tracking calibration profiles. Individual alpha frequency data was also obtained by instructing subjects to close their eyes three times for approximately five seconds at a time.

Data collection and task presentation were performed on the same computer using the Unlock software. EEG and gaze data were streamed into Unlock using LSL. Event markers, such as the start of a c-VEP presentation cycle, were generated in software by Unlock and synchronized with the EEG and eye gaze data streams using the relative timestamps generated by LSL.

All tasks involved attending to stimuli flickering according to one of four possible 31-bit m-sequence-based patterns. A one in the sequence corresponded to the stimulus being on, or visible, while a zero corresponded to the stimulus being off, or hidden. The sequence progressed at a rate of 30Hz, requiring 1.034 seconds to complete one full presentation of the pattern.

In the tasks, four white squares, 180x180 pixels in size (4.7cm, 3.86° visual angle), were centered 360 pixels (9.2cm, 5.81° visual angle to the inner edge) above, below, to the left, and to the right, respectively, from the center of the screen in front of a black background. This arrangement was chosen to reserve sufficient room in the center of the screen for the user application workspace.

The subject was instructed, through an onscreen prompt in the form of an arrow and fixation indicator (a cross) to look at and directly attend to the indicated target. An additional prompt had them look only at the center fixation point as a null-class reference.

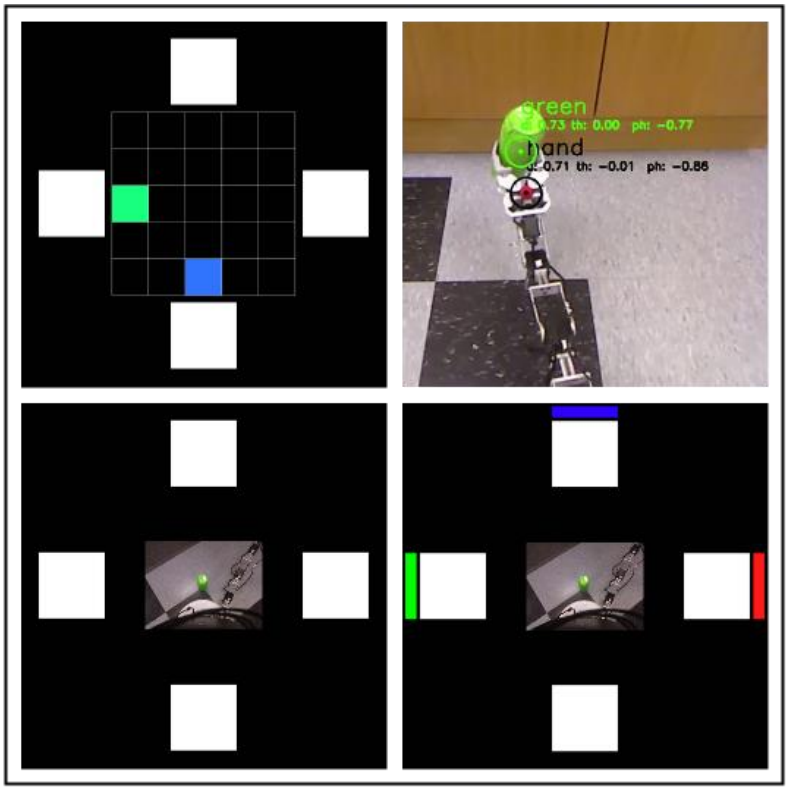

FIGURE 3 : UNLOCK configurations used for robot control and subject acclimitization. Top left: the GridCursor application, used to train subjects on the control paradigm. Top right: A reconstructed view, with overlaid annotations, of the robot arm used in the navigation and grasping task. Bottom left: the manual mode view of the robot control application. Selection of the top and bottom target controls forward and reverse motion; left and right control rotation about the base. Bottom right: autonomous control of the robot. Left (green) and right (red) targets select the colored target to navigate to.

Subjects were presented with the Grid Cursor application first and asked to navigate the blue cursor to the green target then attempt a double-blink selection action. When cursor control appeared reliable, subjects were given a few minutes to reach several targets or to explore movement on their own. When cursor control appeared unreliable, the subject was instructed to ignore the green target and look at a particular stimulus repeatedly instead. If control continued to be unreliable, the session was terminated.

4. Discussion

Unlock has been used successfully in a variety of experimental modalities. A novel use of Unlock was demonstrated with the on-line control of a collaborative robot (co-robot) designed to work with human operators to perform assistive tasks. A critical design challenge in co-robot development is development of efficient decision strategies for how to operate in dynamic environments with autonomous agents (hayes and scassellati, 2013).

The CoCoRo (cognitivie co-robot) system was comprised of an iRobot Create robot base, a 7-degree-of-freedom grasping arm and a Kinect vision sensor, thus having spatial sensing capability. It was controlled via Unlock in both a manual and autonomous mode, and used multiple BCI modalities to guide an adaptive neural network-based control schema that uses exploratory learning to provide autonomous search, navigation, and grasping capabilities for the co-robot to successfully navigate to remote target objects. (Galbraith 2015). The BCI system was primarily configured to receive input through Unlock with the c-VEP EEG method (Galbraith, 2015). Similar tasks to the following were also

demonstrated with eye tracking as the BCI modality, and in the c-VEP case, gaze position was simultaneously tracked with a Tobii EyeX eye tracker.

In manual mode, BCI users were able to perform a navigation task in which the robot was guided towards one of two targets placed within 2m of the robot position. The application screen was a live video feed of the robot field of view. Unlock Commands in this application directed the robot to move either forward or back at a rate of 50 mm/s, or rotate in place to the left or right. A Selection event caused the robot to stop moving.

In automatic mode, BCI users issued Commands to select one of two target objects of different color, and the robot autonomously navigated towards them at a rate of 300mm/s. A Selection event again caused the robot to halt.

5. Conclusion

The Unlock framework is an open source tool for BCI researchers that provides a unified mechanism for designing a BCI task or psychophysical experiment, acquiring data from many kinds of bio-signal acquisition devices, processing them with a high degree of temporal accuracy across multiple devices, and using them to control an easily extended group of applications. By using the Python programming language, Unlock is easy to integrate into a wide variety of other common tools for data analysis, plotting, and result interpretation.

Acknowledgements

This work was supported in part by the Center of Excellence for Learning in Education, Science, and Technology, a National Science Foundation Science of Learning Center (NSF SMA-0835976). Jon Brumberg, Sean Lorenz, Giang Nguyen, James Percent, and Dante Smith also contributed to the development of the Unlock software.

References

For the references cited herein, see the bibliography of:

Galbraith, B.V. (2016) A brain-machine interface for assistive robotic control. *Boston University Thesis. Boston, MA: Boston University.*