Dreamento: An open-source dream engineering toolbox utilizing sleep wearable

Mahdad Jafarzadeh Esfahani [1*], Amir Hossein Daraie [1,2], Frederik D. Weber[1,3], Martin Dresler [1]

[1] Donders Institute for Brain, Behaviour and Cognition, Radboudumc, The Netherlands

[2] Department of Biomedical Engineering, Johns Hopkins University School of Medicine, Baltimore, Maryland, USA

[3] Department of Sleep and Cognition, Netherlands Institute for Neuroscience, an institute of the Royal Netherlands Academy of Arts and Sciences, Amsterdam, The Netherlands

[*] Corresponding author

Abstract

We introduce Dreamento (Dream engineering toolbox), an open-source Python package for dream engineering utilizing the ZMax (Hypnodyne Corp., Sofia, Bulgaria) headband sleep wearable. Dreamento main functions are (1) real-time recording, monitoring, analysis, and stimulation in a graphical user interface (GUI) (2) and offline post-processing of the resulting data. In real-time, Dreamento is capable of (1) recording data, (2) visualizing data, including power-spectrum analysis and navigation, (3) automatic sleep-scoring, (4) sensory stimulation (visual, auditory, tactile), (5) establishing text-to-speech communication, and (6) managing the annotations of automatic and manual events. The offline functionality aids in post-processing the acquired data with features to reformat the wearable data and integrate it with non-wearable recorded modalities such as electromyography. While the primary application of Dreamento was developed for (lucid) dreaming studies, it is open to being adapted for other purposes and measurement modalities.

Keywords: wearable, sleep, Python, open-source, lucid dreams, dream engineering, EEG, PSG

Introduction

The gold standard to measure human sleep is polysomnography (PSG) which consists of electroencephalography (EEG), electromyography (EMG), and electrooculography (EOG) as the primary physiological signals. The American Academy of sleep medicine (AASM) also recommends recording additional modalities such as electrocardiography (ECG), blood oxygen saturation level, and body position while studying sleep (Iber, 2007). PSG recordings are, however, accompanied by several constraints, including limitations to artificial lab environments and the time, effort, and thus costs to place the electrodes. Also, standard sleep scoring based on the PSG recordings is time-consuming and subject to considerable variability in inter-rater agreement.

Based on advancements in miniature electronics, various wearable systems such as smartwatches, smart rings, and EEG headbands have recently been launched in the consumer technology market. Among these, given the prominence of scalp EEG in studying sleep, EEG headbands have received substantial attention from sleep researchers. While wearable systems overcome some limitations of PSG, they also have restrictions. Headbands utilize an EEG montage different from AASM standards which makes manual scoring more of a challenge. Nonetheless, automated sleep scoring, when validated, can help alleviate those challenges. Also, many EEG headbands process the data *onboard* which also hinders heavy computations. Accordingly, onboard computations such as automatic sleep scoring and sleep modulation are in most cases not very reliable. Less affected by these constraints are wearable headbands with either cloud computing features or the capability to communicate with a computer in real-time, enabling extensive processing feasible through the use of computer resources. The ZMax (Hypnodyne Corp., Sofia, Bulgaria) sleep wearable is an example of an EEG headband with a real-time data transmission feature that discloses a transmission control protocol/internet protocol (TCP/IP) socket to parse the data. This gives considerable freedom to software developers to design software for a variety of purposes and consequently makes the performance of wearable systems more reliable.

To serve as a reliable alternative to PSG at least for some application cases, supplementary analysis tools are needed so that the output of wearables can make up for their shortcomings compared to PSG devices. Several open-source sleep analysis toolboxes are available, e.g., tools to visualize and analyze sleep data such as SleepTrip (RRID: SCR_017318, https://github.com/Frederik-D-Weber/sleeptrip), Sleep (Combrisson et al., 2017), Visbrain (Combrisson et al., 2019), YASA (Vallat & Walker, 2021), and various open-source automatic sleep scoring algorithms (e.g. Perslev et al., 2021; Supratak et al., 2017; Supratak & Guo, 2020; Vallat & Walker, 2021). The current research, however, lacks open-source tools to monitor, analyze, and modulate sleep in *real-time*. This brought us to develop an open-source dream engineering toolbox with unique features.

Exploiting the existing features of the ZMax headband, we developed an open-source, Python-based toolbox for dream engineering, dubbed Dreamento (Dream engineering toolbox, https://github.com/dreamento/dreamento), to record, monitor, analyze, and modulate sleep in real-time as well as for offline post-processing. By introducing Dreamento, we intend to facilitate sleep and dream research so that we can provide a standard tool for performing experiments with minimal sensing systems, in a real-life environment, and with large sample sizes. Among the most notable features of Dreamento in real-time are (1) data recording, (2) data visualization comprising power-spectrum analysis and navigation (3) automatic sleep scoring, (4) sensory stimulations (visual, auditory, tactile), (4) text-to-speech communication, and (5) saving annotations of the automatic and manual events. To propose an all-in-one package, Dreamento is also capable of post-processing the acquired data with ZMax headband (e.g., EEG, acceleration) and integrating it with the resulting data recorded by other measurement modalities such as EMG.

Methods

Programming language, dependencies

Dreamento was implemented in Python, as an easy-to-learn programming language that is widely used in the field with stable open-source software packages as a basis to build on and upon. A detailed list of all the dependencies of Dreamento on external libraries can be found on the Dreamento Github page. To install our package, the user should employ Conda (https://conda.io), as an open-source environment manager to create a virtual environment based on the required dependencies (instructions can be found on the Github page) to run Dreamento. Our package is developed and tested on a Windows computer with 16 GB physical memory (also known as random-access memory - RAM) but is also compatible with macOS, and Linux. Although we developed Dreamento in Python, due to a large number of MATLAB (Mathworks, Natick, Massachusetts, USA) users, we have also given the possibility to export all the raw and processed data from Dreamento into MATLAB.

Hypnodyne software suite

The producer of the ZMax headband, Hypnodyne, provides a software suite including HDFormat, HDScorer, HDServer, and HDRecorder (which can be freely downloaded from the official website of the company, https://hypnodynecorp.com). For the purpose of real-time recording or representation, the HDServer initiates the TCP/IP server and HDRecorder operates as the main client of the server capable of recording various signals such as two EEG channels, triaxial acceleration, photoplethysmography (PPG), body temperature, ambient noise, ambient light, and battery voltage. Nevertheless, Some functionalities which are practical for (lucid) dream engineering studies were not supported by HDRecorder. In real-time recording, (1) the time and amplitude axes should be adjustable (e.g., to set up the desired amplitude differently while detecting eye movements during rapid eye movement (REM) sleep versus detecting spindles in N2 sleep), (2) information regarding the sensory stimulation such as the stimuli type and the exact time of presentation should be automatically kept, (3) online autoscoring potentially supporting various algorithms should be available to assist the experimenter with real-time scoring of data, (4) it should be possible to mark the desired annotations once a remarkable event happens, and (5) additional signal qualities, e.g., power-spectrum analysis with a time-frequency representation (TFR) should be provided as complementary information for online scoring and analysis of sleep.

Program structure

As shown in table 1, Dreamento comprises different programming classes, namely *ZmaxSocket*, *ZmaxDataID, ZmaxHeadband*, *Window*, *RecordThread, and OfflineDremento*. We defined the configurations of connection to the TCP/IP server (e.g., host IP address and the port number) in *ZmaxSocket*. In addition, this class is responsible for establishing two-way communication between the client and the server, i.e., data chunk transmission from the server to the client and sending commands/messages such as stimulation properties from the client to the server. To enhance the code's readability, *ZmaxDataID* enumerates the possible data to be collected (e.g., EEG channels and triaxial accelerometer) with a specific identity number. The choice of which data to record (e.g., EEG channels only or together with acceleration) in addition to the initialization of the buffer sizes for each data channel were incorporated in the *ZmaxHeadband* class. The relevant data measures (e.g., deriving the correct EEG, temperature, and acceleration values from the raw measurements) were set in this class as well. This class is also meant to send messages to the server utilizing *ZmaxSocket* (e.g., stimulation commands) and thus include the corresponding hexadecimal to decimal converter.

Programming class	Function
ZmaxSocket	Establishing a connection to the TCP/IP server
ZmaxDataID	Enumerating each data signal with a specific identity number
ZmaxHeadband	Receiving data buffer and sending stimulation commands
Window	Data representation, analysis, and defining the GUI functionalities
RecordThread	Transmitting data to the main window when ready for plotting and analysis
OfflineDremento	Post-processing of the recorded data

Table 1 - *The main functions of different programming classes used in the development of Dreamento.*

All variables related to the data recording (e.g., which signals to record), monitoring (e.g., time and amplitude scales of signals), analysis (e.g., activation of real-time autoscaling), and stimulation (stimulus properties) are specified in the *window* class. This class also determines the functions associated with all the GUI buttons, from a primary button that activates the connection to the server, to the ones triggering stimulation commands. We designed *RecordThread* as a thread to fetch the data in real-time and send it to the *window* as well as to maintain the accurate timing of the processes. This was done so that all other features of the toolbox, such as stimulation or setting markers, can be done while displaying data and other processes do not get frozen. The thread keeps track of the received number of samples to the server (as a measure of the passed time) and once an epoch of 30 seconds (equivalent to 7680 samples based on 256 Hz sampling rate) is over, the thread indicates that the buffer to analyze the data is ready and subsequently send it to the *window*. The rest of the analysis, namely the autoscoring, spectrogram, and periodogram analysis then occurs in the *window* class. All post-processing functions of resulting data are done by *OfflineDremento* class.

Graphical user interface (GUI)

Figure 1 shows the end-user pipeline of Dreamento. The ZMax EEG headband is built in such a way that the server only allows communication with another client (i.e., Dreamento) if the main client (i.e., HDRecorder) has already been connected. Therefore, after running the HDserver (the TCP/IP server), the HDRecorder should be first introduced to the server, and only then Dreamento client is able to connect by clicking on the ‘connect’ button. Once the connection to the server is established, the recording can be started by clicking the ‘record’ button.

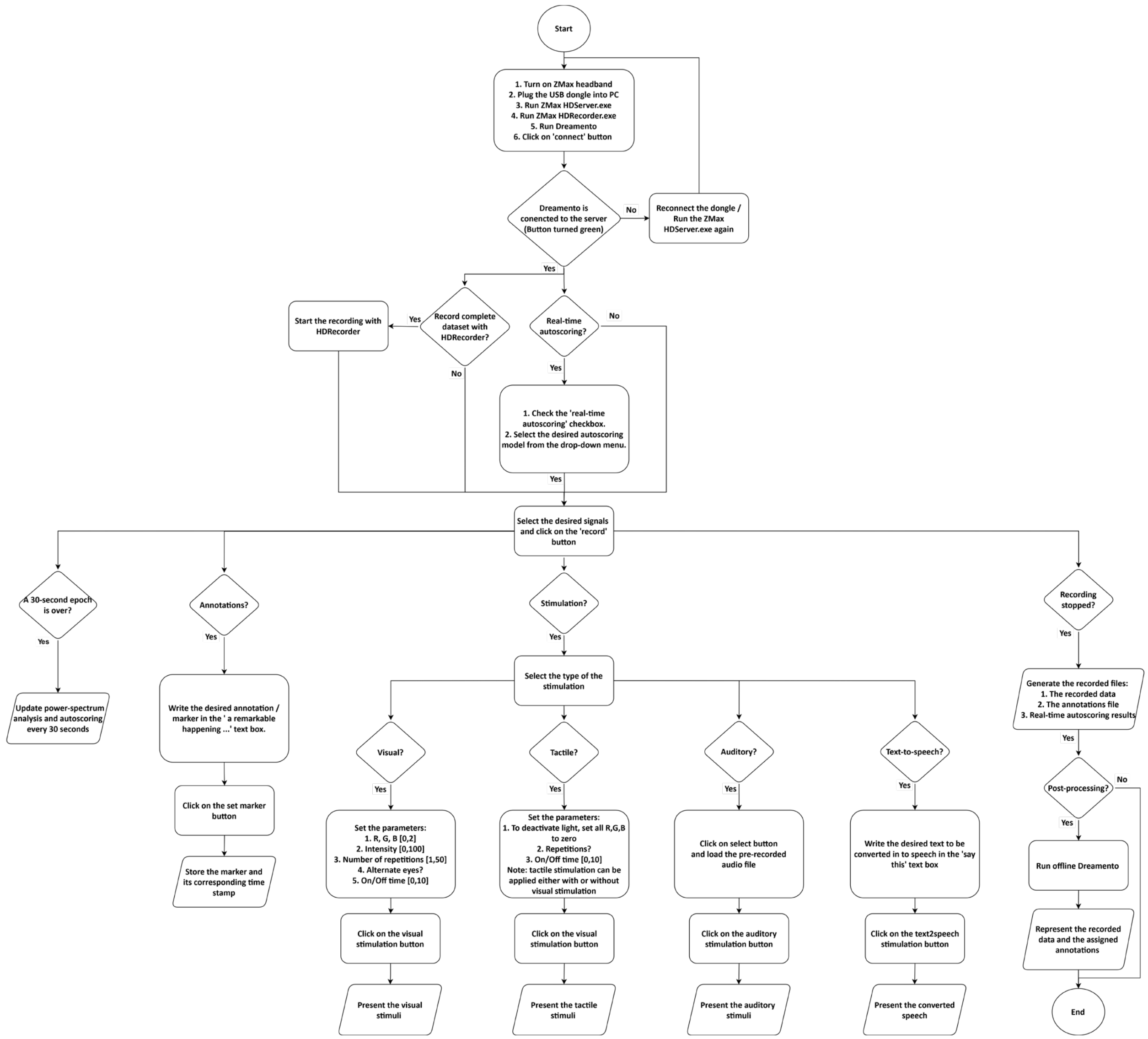

Figure 1 *- Dreamento end-user pipeline to record, monitor, analyze, and stimulate sleep in real-time.*

While recording, the software depicts real-time EEG signals with adjustable scales for time and amplitude axes (see figure 2, bottom panel). The stimulation panel located on the top left side of figure 2 has the corresponding parameters for the visual, auditory, tactile, and text-to-speech stimulation. For instance, for visual stimulation, one can set up a desired color of the light (by combining red, green, and blue colors), choose the number of on/off repetitions, determine whether the two light-emitting diodes (LEDs) of the headband should turn on and off simultaneously or alternatively, select the required intensity of light, and opt the on/off time of LED. As shown in figure 2, the real-time analysis panels consist of the spectrogram located on the top right, the periodogram located in the middle right, and the autoscoring prediction panel located in the middle of the GUI. The autoscoring and periodogram keep the values for the last 30-second epoch only, whereas the spectrogram maintains the output from the last four epochs (two minutes) for the user to have an estimate of the recent sleep stage transitions.

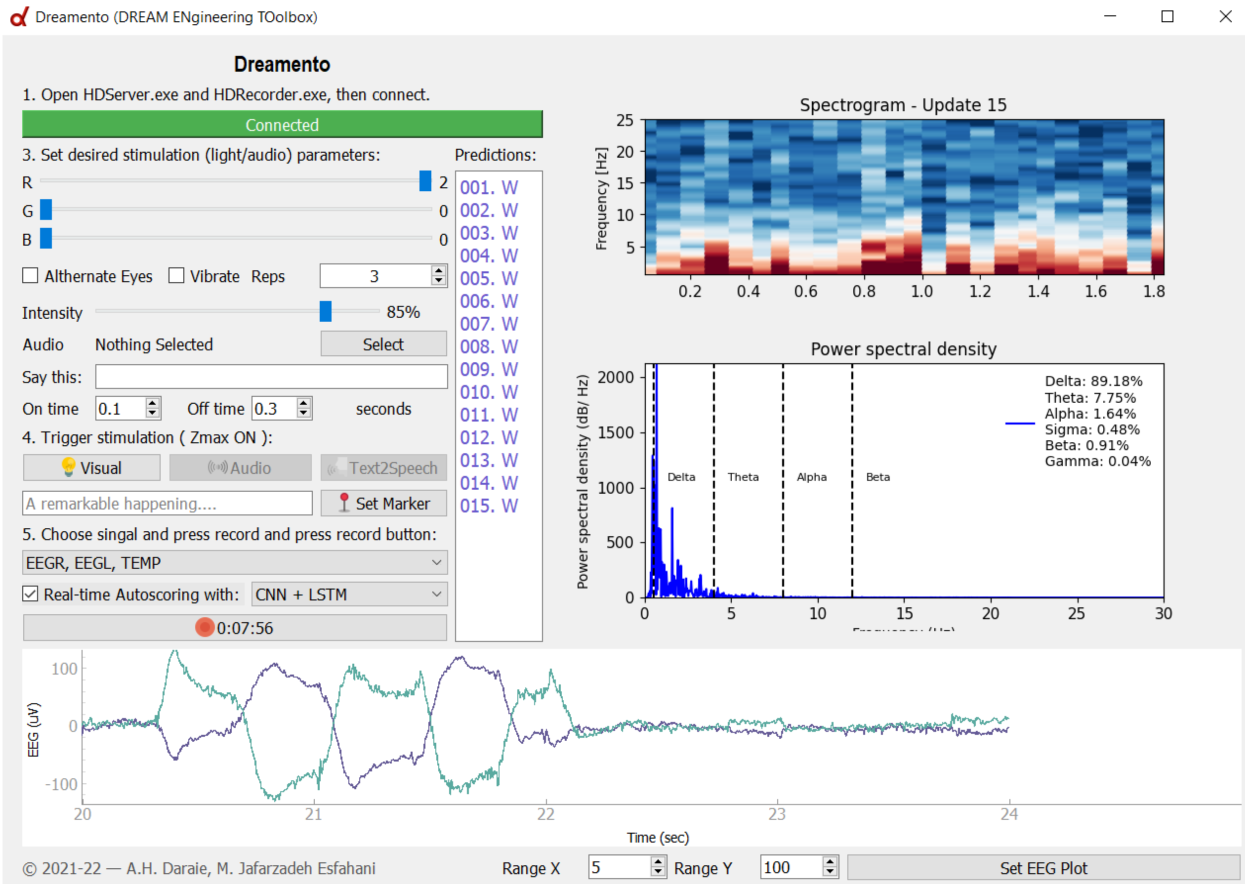

***Figure 2** - GUI of real-time Dreamento.*

By stopping the data recording, the software creates three files, namely, the actual recorded data (.txt), annotations (.json), and real-time scoring (.txt) results. Given the inconsistency of the sampling rates during wireless communication (the fluctuations from the actual 256 Hz sampling rate of the headband), the recorded data file stores not only the data points but, the number of transmitted samples per second. This way, one can always count the actual time. The annotation file stores all the manually set markers and the stimulation annotations together with all the assigned properties and the actual time stamp.

In case the end-user is interested in storing only a subset of the data that the headband is able to record (e.g., EEG channels, and acceleration only, without additional information such as ambient noise or temperature), the recording can be done through the Dreamento only. On the other hand, if all data that ZMax headband can record is of interest, the user should start the recordings both with the HDRecorder and Dreamento software (see figure 1). In the latter case, which is recommended by us, due to the time difference (albeit short) at the start of recording between the two programs, a data synchronization procedure is required. To do this, the user should employ Dreamento which utilizes the recorded data by the HDRecorder (comprising all the data that the headband is able to record, e.g., EEG, acceleration, microphone recordings), the Dreamento data file, Dreamento annotation files, and optionally the data recorded by another measurement modality, e.g., EMG recordings. The synchronization process is based on a cross-correlation analysis to find the lag between the starting time stamps of the data recorded by HDRecorder and Dreamento resulting data (see data synchronization section for details).

Offline Dreamento integrates different sources of data/information and represents them in a GUI (figure 3). The top three rows of the window (figure 3 – panel A) are assigned to the annotations, stimulation markers, and TFR of the whole recording. Thus, with a glance at the first three rows of the display, the user gets an overview of the annotation distributions, stimulation types and timing (shown with red, blue, and green for visual, auditory, and tactile stimulations, respectively), as well as an estimate of sleep stage transitions using the TFR. All the rest of the rows (panels B to D) correspond to the single epoch determined by the black vertical line shown in the overall TFR (figure 3, A-3). These rows represent the annotations (figure 3, B-1), stimulation markers (figure 3, B-2), triaxial acceleration (figure 3, B-3), ambient noise (figure 3, B-4), three EMG channels (figure 3, panel C), TFR (figure 3, D-1), and two channels of EEG (figure 3, D-2) of the selected epoch. For a better mapping between the annotation descriptions and their corresponding time stamp, a specific color and a number are assigned to each of them.

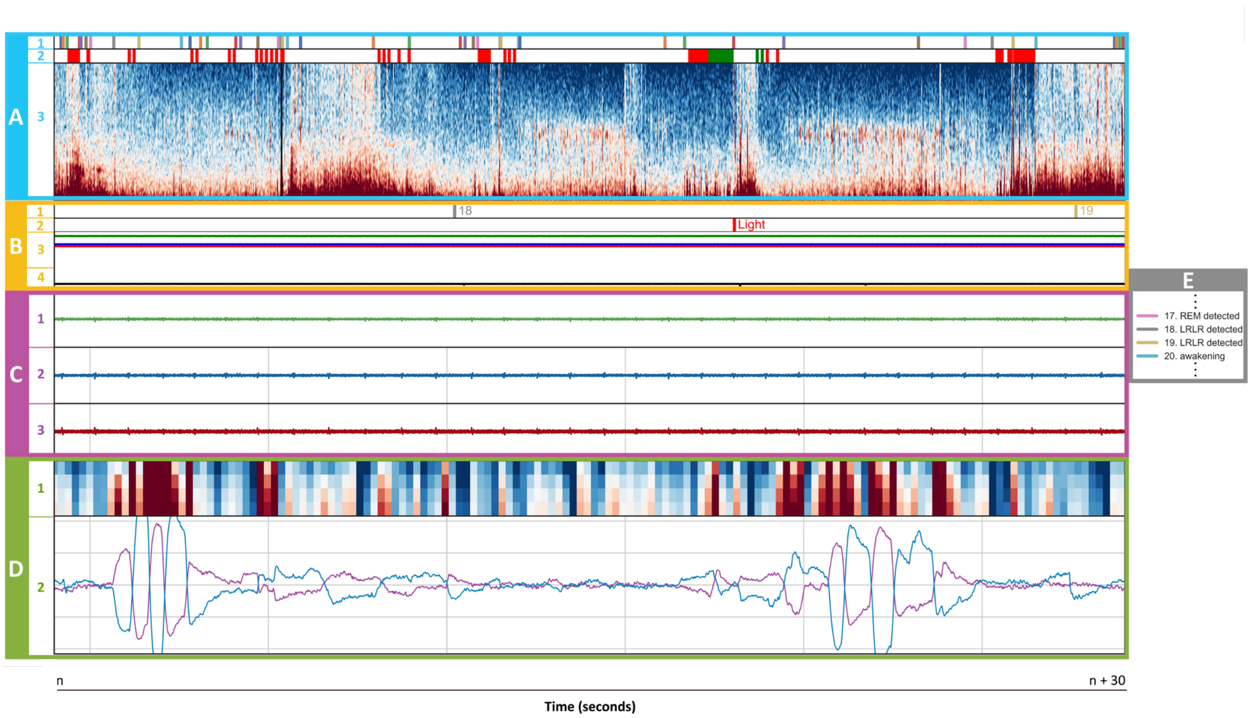

Figure 3 *- GUI of offline Dreamento (units, values, and labels on the vertical axes are removed for a clearer representation). Panel (A) depicts the overall recording representation of (1) annotations: each number corresponds to an annotation, (2) stimuli presentation (red: visual, green: tactile, blue: auditory stimuli), and (3) corresponding TFR. The black line in the TFR shows the currently selected epoch. Panels B to D belong to the current epoch, as indicated in the bottom, from the second n to n + 30. Panel B shows the distribution of the (1) annotations, (2) stimuli presentation, (3) triaxial acceleration indicated by red, green, and blue, and (4) ambient noise (the flat black line represents no ambient noise/sound). Panel C shows three EMG channels recorded by another measurement modality, but, integrated into Dreamento for post-processing. The EEG representation of the currently selected epoch together with its corresponding TFR is shown in panel D. The annotations are shown in panel E.*

Data synchronization

While recording data simultaneously with Dreamento and HDRecorder, it is not possible to assure that the start point of recording in both software has exactly the same time stamp. Thus, we developed a synchronization algorithm to automatically align the recordings (figure 4). The synchronization process starts by first loading EEG data recorded by both Dreamento and HDRecorder . Then, Dreamento selects a portion of the recorded data (e.g., from 100 - 130 seconds as in default settings) and subsequently applies a cross-correlation analysis. Based on the cross-correlation results, Dreamento finds the lag corresponding to the maximum amplitude of the cross-correlation function and shifts the signals to fully synchronize them. Eventually, the data recorded by HDRecorder such as EEG, ambient noise, and acceleration are all fully aligned with the recordings from Dreamento, e.g., EEG, the manual and automatic annotations, and the stimulation information.

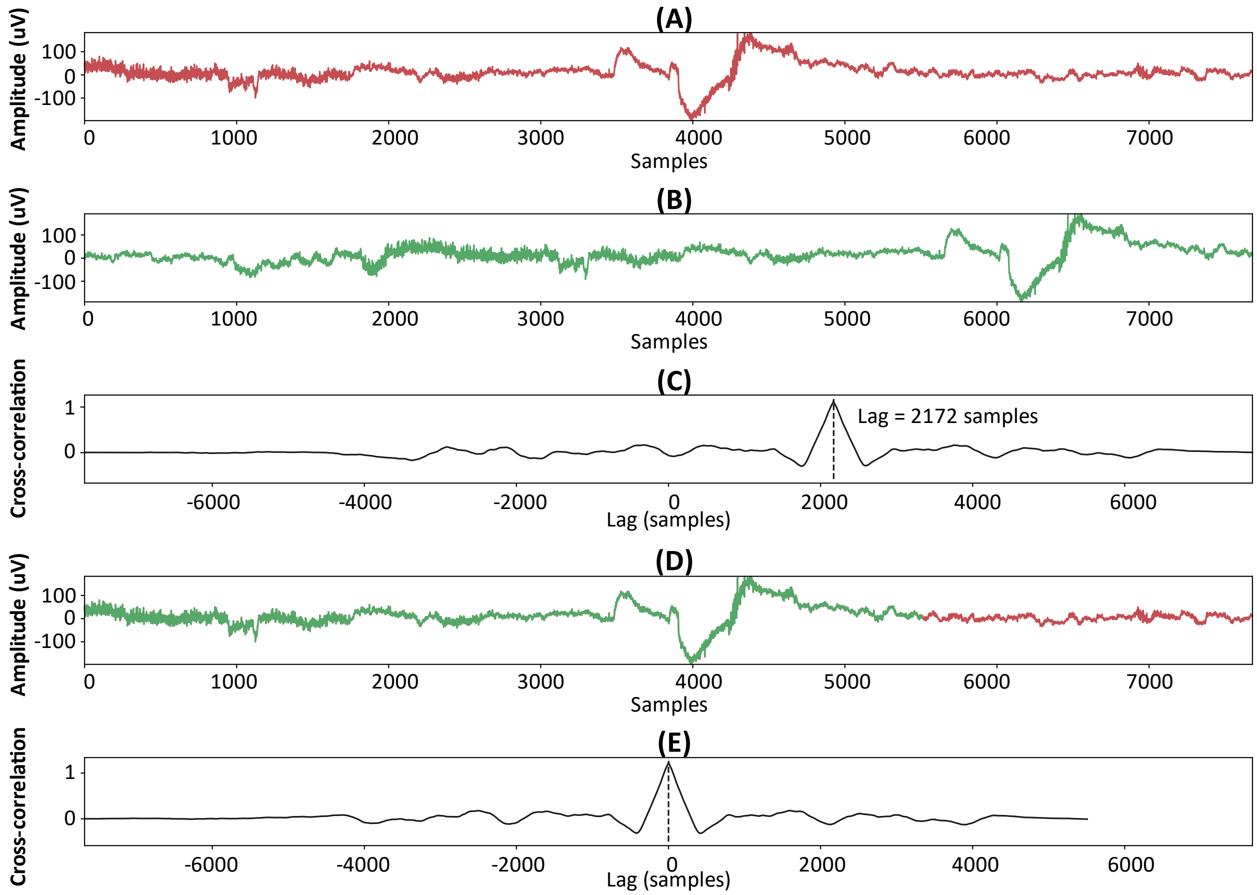

__Figure 4__ - Synchronization procedure during post-processing. First, a sample epoch of 30 seconds (i.e., 7680 samples) from the recorded (A) Dreamento (red signal) and (B) HDRecorder (green signal) was chosen. (C) Cross-correlating the signals in (A) and (B) resulted in a peak in the cross-correlation function which corresponded to the lag (2172 samples in this case) between the signals. (D) Dreamento and HDRecorder signals plotted on top of each other after the synchronization based on the cross-correlation results and (E) proof of no lag (i.e., full alignment) between the signals at the end of the synchronization process.

Documentation

Our toolbox is delivered with a detailed and step-by-step documentation, from how to install, activate and utilize the package for the end-users, to the detailed description of different programming classes, methods, and functions useful for developers. This way, we tried to facilitate the contribution of both the software developers and the end-users to utilize Dreamento. The relevant documentation for Dreamento can be found online at https://dreamento.github.io/docs/.

Discussion

In this paper, we introduced Dreamento as a dream engineering toolbox capable of real-time recording, monitoring, analysis, and modulation of sleep. To date, the majority of (lucid) dreaming research had sensible limitations such as lack of generalizability (constrained to the geographical boundaries), validity (absence of physiological measurements and merely taking the self-reports), and low power (small sample sizes). By developing Dreamento, we aimed at simplifying sleep and (lucid) dreaming research with wearable systems to overcome these limitations.

A prospective study can explore the efficiency of various lucid dreaming induction methods using Dreamento, in different locations and on a larger scale. Moreover, using our toolbox, the applications of establishing two-way communication between the (lucid) dreamer and the outer world (e.g., memory and learning) can be explored (Konkoly et al., 2021). While Dreamento was actually designed for (lucid) dream engineering purposes, its applications should not be limited to these. The toolbox has features that can be used for any modulation of either REM or non-REM sleep. Intriguingly, any topic in sleep research that requires sensory stimulation, e.g., targeted memory reactivation (TMR), can employ Dreamento.

Real-time analysis typically comes with some challenges. Importantly, the real-time analysis algorithms should be very accurate to perform the desired analysis accurately, and on the other hand, they should be performed very quickly so as not to create a disruption in the synchronization of the data stream. In our study, the real-time analysis comprises autoscoring, spectrogram, and periodogram updates after every epoch of 30 seconds. This means that every 30 seconds, our program does not receive new input from the server for a very short period of time (for example, a few milliseconds) so that it can apply the relevant analysis to the data received during the previous 30 seconds. While the program is busy with the real-time analysis and thus closes the new data entry gateway, the data sent from the server remains in the queue to be entered into the program as soon as the analysis is finished. This means that if we simply let the queued data enter (which will be accumulated over time) the software, the program will no longer work in actual real-time and therefore works with some time jitter. To solve this problem, Dreamento automatically ignores the very small portion of the data that remains in queue during the real-time analysis and therefore has always synchrony with the real-time data received from the server, regardless of the duration of the recording.

Various offline automatic sleep scoring algorithms have been proposed in recent years. That is, however, not the case with the real-time automatic scoring of sleep. Real-time autoscoring (after every 30-second epoch) should be accurate enough to reliably detect the sleep stage of interest and at the same time light enough to be fast and not seriously interfere with other functions of the program. In this study, we employed TinySleepNet (Supratak & Guo, 2020) as the default autoscoring algorithm trained on a dataset collected from a citizen neuroscientist who measured his nocturnal sleep with simultaneous ZMax and PSG (paper in preparation). Nonetheless, We intend to improve the performance of this algorithm over time.

Acknowledgments

This work was supported by the PPP Allowance made available by Health~Holland.

Author Contributions

Conceptualization: MJE; Methodology: MJE, AHD; Software: AHD, MJE; Validation: MJE, AHD; Formal analysis: MJE, AHD; Investigation: AHD, MJE; Resources: MJE, AHD; Data curation: MJE, AHD; Writing – original draft: MJE; Writing – review & editing: MJE, AHD, FDW, MD; Supervision: FDW, MD; Project administration: MJE, MD; Funding acquisition: MD.

References

Combrisson, E., Vallat, R., Eichenlaub, J. B., O’Reilly, C., Lajnef, T., Guillot, A., Ruby, P. M., & Jerbi, K. (2017). Sleep: An open-source python software for visualization, analysis, and staging of sleep data. *Frontiers in Neuroinformatics*, *11*. https://doi.org/10.3389/fninf.2017.00060

Combrisson, E., Vallat, R., O’Reilly, C., Jas, M., Pascarella, A., Saive, A. L., Thiery, T., Meunier, D., Altukhov, D., Lajnef, T., Ruby, P., Guillot, A., & Jerbi, K. (2019). Visbrain: A multi-purpose GPU-accelerated open-source suite for multimodal brain data visualization. *Frontiers in Neuroinformatics*, *13*. https://doi.org/10.3389/fninf.2019.00014

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

Iber, C. (2007). The AASM manual for the scoring of sleep and associated events: Rules. *Terminology and Technical Specification*.

Konkoly, K. R., Appel, K., Chabani, E., Mangiaruga, A., Gott, J., Mallett, R., Caughran, B., Witkowski, S., Whitmore, N. W., Mazurek, C. Y., Berent, J. B., Weber, F. D., Türker, B., Leu-Semenescu, S., Maranci, J. B., Pipa, G., Arnulf, I., Oudiette, D., Dresler, M., & Paller, K. A. (2021). Real-time dialogue between experimenters and dreamers during REM sleep. *Current Biology*, *31*(7), 1417-1427.e6. https://doi.org/10.1016/j.cub.2021.01.026

Perslev, M., Darkner, S., Kempfner, L., Nikolic, M., Jennum, P. J., & Igel, C. (2021). U-Sleep: resilient high-frequency sleep staging. *Npj Digital Medicine*, *4*(1). https://doi.org/10.1038/s41746-021-00440-5

Supratak, A., Dong, H., Wu, C., & Guo, Y. (2017). DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *25*(11), 1998–2008. https://doi.org/10.1109/TNSRE.2017.2721116

Supratak, A., & Guo, Y. (2020). *TinySleepNet: An Efficient Deep Learning Model for Sleep Stage Scoring based on Raw Single-Channel EEG; TinySleepNet: An Efficient Deep Learning Model for Sleep Stage Scoring based on Raw Single-Channel EEG*. https://doi.org/10.0/Linux-x86_64

Vallat, R., & Walker, M. P. (2021). An open-source, high-performance tool for automated sleep staging. *ELife*, *10*, e70092. https://doi.org/10.7554/eLife.70092