Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 July 2021 doi:10.20944/preprints202107.0651.v1

*Research Article*

# OpenSync: An opensource platform for synchronizing multiple measures in neuroscience experiments

Moein Razavi [1,2], Vahid Janfaza [2], Takashi Yamauchi [1*], Anton Leontyev [1], Shanle Longmire-Monford [1], Joseph Orr [1]

[1] Department of Psychological and Brain Sciences, Texas A&M University
[2] Department of Computer Science and Engineering, Texas A&M University
* Correspondence: moeinrazavi@tamu.edu

**Abstract:**
*Background:* The human mind is multimodal. Yet most behavioral studies rely on century-old measures such as task accuracy and latency. To create a better understanding of human behavior and brain functionality, we should introduce other measures and analyze behavior from various aspects. However, it is technically complex and costly to design and implement the experiments that record multiple measures. To address this issue, a platform that allows synchronizing multiple measures from human behavior is needed.
*Method:* This paper introduces an opensource platform named OpenSync, which can be used to synchronize multiple measures in neuroscience experiments. This platform helps to automatically integrate, synchronize and record physiological measures (e.g., electroencephalogram (EEG), galvanic skin response (GSR), eye-tracking, body motion, etc.), user input response (e.g., from mouse, keyboard, joystick, etc.), and task-related information (stimulus markers). In this paper, we explain the structure and details of OpenSync, provide two case studies in PsychoPy and Unity.
*Comparison with existing tools:* Unlike proprietary systems (e.g., iMotions), OpenSync is free and it can be used inside any opensource experiment design software (e.g., PsychoPy, OpenSesame, Unity, etc., *https://pypi.org/project/OpenSync/* and *https://github.com/moeinrazavi/OpenSync_Unity*).
*Results:* Our experimental results show that the OpenSync platform is able to synchronize multiple measures with microsecond resolution.

**Keywords:** multiple measures synchronization; automatic device integration; open-source; PsychoPy; Unity

---

## 1. Introduction

### *1.1. Problem Statement*

The brain supervises different autonomic functions such as cardiac activity, respiration, perspiration, etc. Current methods to study human behavior include self-report, observation, task performance, gaze, gait and body motion, and physiological measures such as electroencephalogram (EEG), electrocardiogram (ECG), functional magnetic resonance imaging (fMRI). Human behavior is inherently multimodal and interdependent; however, most of the studies that provided useful information in the past used only a single measure and did not consider the link between different measures [1]. Another important reason to consider multimodal experiments is that measures from different sources have location (spatial) or timing (temporal) overlap in the brain [2]. Hochenberger (2015) suggests that the experiments that use multiple measures, facilitate the perception of senses that operate in parallel [2]. Hence, to distinguish between different functions in the mind and to have a more accurate interpretation of the behavior, using multiple measures is more informative than a single measurement [4]. In other words, in a similar way that using multiple features improves the

 doi:10.20944/preprints202107.0651.v1

accuracy in a classification task, combining different measures helps to improve the predictions of human behavior in the experiments that study brain functionality and its association with behavior [5].

Despite the need for multimodal experiments, very few systems exist to integrate different measurements. The problem is that integrating multiple measurements is quite complicated. There are several proprietary and non-proprietary systems (e.g., iMotions [6], BioPac [7], OpenViBE [8], BCI2000 [9], etc.) that ease the implementation of the experiments that employ multiple measures; however, these systems have several drawbacks. First, it is costly and challenging to integrate different devices. Second, the stimulus presentation and data acquisition should be in different software which makes it difficult for synchronization between multiple data streams (since different software may have different processing times which results in undesirable delays). Third, the devices should be connected and started one by one which is time-consuming while instant and easy experiment setup is essential for collecting data from a large group of participants. Finally, storage and accessibility of the data from multiple measurements in a format suitable for data analysis, is difficult.

In this paper, we present OpenSync, a system that allows stimulus presentation, data acquisition and recording in the same software and how to make this process automatic and adaptable for various experiments. Next section describes the features of OpenSync platform that help with creating reliable multimodal experiments.

### *1.2. Our Contribution*

Below, we present the criteria we considered in the development of OpenSync to make it useful for human behavior research experiments.

- **Multi Device Synchronization –** Synchronize several sensors, I/O devices and markers with *microsecond* resolution
- **Easy implementation –** the package provides a set of libraries and customized device SDKs that can be imported in psychological experiment design software. Only one computer is needed for device connection, running experiments and recording data.
- **Quick implementation –** integration of multiple measures from multiple sources can be done in a short time without any programming knowledge.
- **Practical –** this software package addresses key challenges in synchronizing multiple sensors for psychological experiments such as being time consuming to setup, synchronize, and record data from multiple resources.
- **Modular –** the data from all the devices and markers can be recorded independently using the individual functions in the proposed platform.
- **Comprehensive –** data acquisition and stimulus presentation can be in the same software.
- **Portable –** can be used within opensource psychological experiment design software (e.g., PsychoPy, OpenSesame, Unity).
- **Automatic –** recording data from multiple sources and saving them on the disk will be done automatically.
- **Adjustable –** the platform is expandable to support new devices and update the current devices' SDKs. Also, all opensource devices, as well as non-opensource devices that can stream data with one of the opensource platforms (e.g., C, C++, C#, Python, Java and Octave), can be synchronized.
- **Open source and Cost-efficient –** this is a free, opensource software package and does not need any extra intermediate hardware.
- **Multiplatform –** this software package is not dependent on operating system, since it has been developed by opensource platforms (i.e., Python, C++, and C#)
- **Offline data acquisition –** prerecorded data can be loaded instead of the data from actual devices. Offline data acquisition is not in the scope of this manuscript and we leave it as a future work.

This paper is structured in the following flow. Section 2 introduces the method we used in developing the OpenSync platform. Section 3 includes two case studies of using OpenSync in PsychoPy and Unity. In Section 4, we test the applicability of OpenSync via time synchronization test. The final section of the paper includes a review of experiments with multiple measures and compares OpenSync with available platforms, followed by summary and possible future extensions for OpenSync.

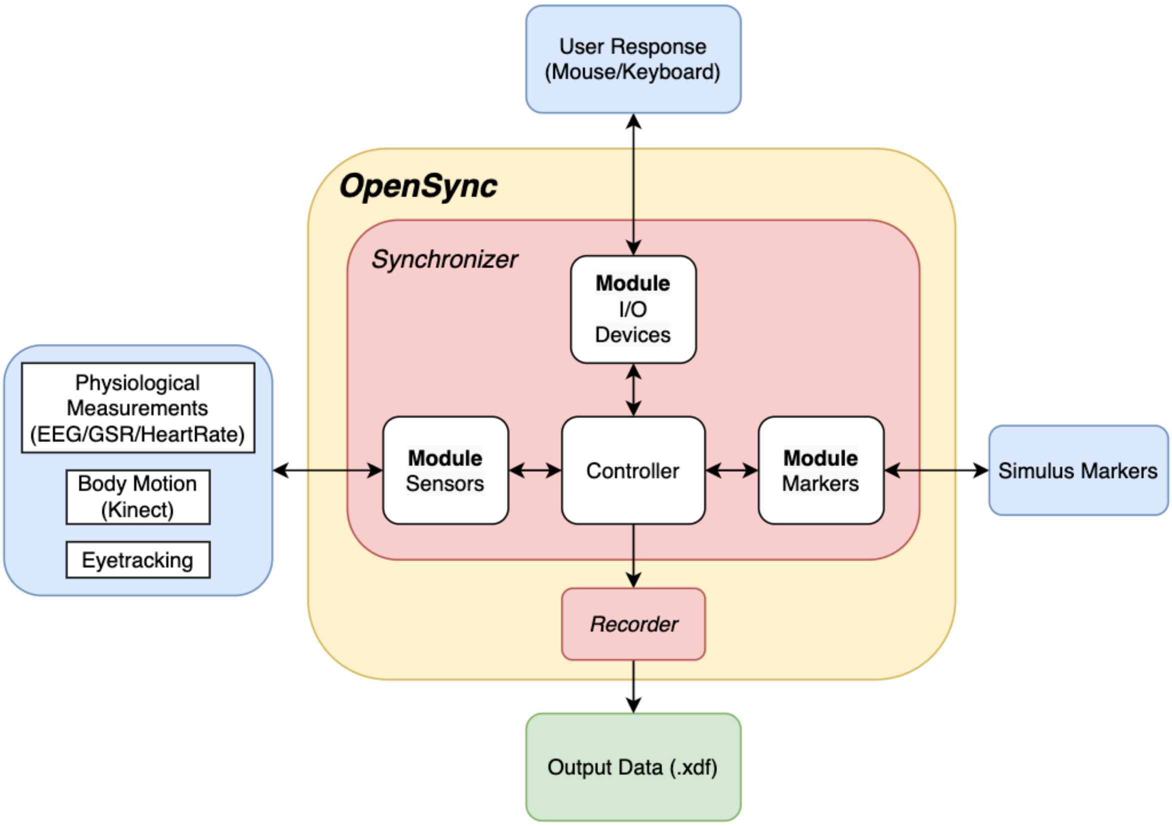



Figure 1. Overall view of OpenSync platform.

## 2. Method

*OpenSync platform*

In this section, we provide the details of the OpenSync platform. Figure 1 shows the overall view of OpenSync platform. OpenSync is a modular platform that can be used to automatically synchronize multiple measures in psychological experiments. It includes one main module, that is used to initialize, synchronize and record data streams. The main module contains two submodules: Synchronizer and Recorder modules. Synchronizer module is the core of the platform which contains Controller, input/output (I/O) module, Markers module, and Sensors module. Controller is responsible for initialization and synchronization of data streams and sending data to Recorder module. It uses Lab Streaming Layer (LSL) as its core protocol for data streaming and synchronization [10]. I/O module is used for streaming user response data from I/O devices (Keyboard, Mouse, Joystick, etc.). Markers module is used for streaming the stimulus presentation markers. Finally, Sensors module is used for streaming data from various biological sensors (EEG, GSR, eye-tracking, body motion, etc.). The Recorder module records all the streams from different submodules in a single file with Extensible Data Format (*.xdf*) extension. Table 1 describes the pseudocode for the overall structure of OpenSync.

 doi:10.20944/preprints202107.0651.v1

Table 1. Pseudocode for the overall structure of OpenSync

**OpenSync Platform**

```
1   # Include required libraries (e.g., LSL)
2   # Include required device SDKs
3   Module OpenSync():
4       _Config_ = Get User Input Parameters
        // Synchronizer module (Lines 5-7)
5       Call class I/O device from module I/O devices (_Config_)   // Call for each I/O device
6       Call class Stimulus Markers from module markers (_Config_)
7       Call class Physiological Sensors from module sensors (_Config_)   // Call for each physiological sensor
8       Initialize XDF Recorder (_Config_)   // Recorder module
9       Start Synchronization and Recording Data
10  Module I/O Devices:
11      Class I/O device:   // Contain class Keyboard, class Mouse, class Joystick, class Gamepad
12          Def __init__(_Config_):
13              Create and initialize I/O device data stream
14          Def stream_data(I/O object):
15              Stream I/O data
16  Module Markers:
17      Class Stimulus Markers:
18          Def __init__(_Config_):
19              Create and initialize stimulus marker data stream
20          Def stream_marker(marker):
21              Stream stimulus marker (int/string)
22  Module Sensors:   // Contains libraries of different physiological sensors
23      Class Physiological Sensors:   // Contains class EEG, class BodyMotion, class EyeTracking
24          Def __init__(_Config_):
25              Initialize physiological sensor and stream data
```

OpenSync platform contains several functions that can be found in Table 2. In this table, functions 2-6 are associated with the initialization, configuration and streaming of EEG devices. For OpenBCI Cython device, if the Daisy module is installed, we can set daisy=True to record the data from Daisy as well. We should specify the USB port that the OpenBCI Bluetooth dongle is connected to which is "COM3" by default. In BrainProducts LiveAmp device, the number of EEG channels (n_channels) can be set on {16, 32, 64} and the sampling frequency (sfreq) can be set on {250, 500, 1000}.

Function 10 runs Gazepoint Control program and streams Gaze data. If the Gazepoint biometric device is connected, the biometrics data can be streamed by setting biometrics=True.
In function 12, by setting clickable_object, pos, and click parameters, we can initialize streams for the name of the clicked object, the mouse position coordinates and the name of the mouse button clicked (left, middle, or right), respectively.

In function 13, by setting keypress and pos parameters, we can initialize streams for the name of the pressed key and the joystick position coordinates, respectively.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 July 2021 doi:10.20944/preprints202107.0651.v1

Table 2. Functions associated with OpenSync platform

| | Functions for Physiological Sensors | Description |
|---|---|---|
| 1 | **EEG = OpenSync.sensors.EEG()** | Initialize, configure and stream EEG devices data |
| 2 | EEG.OpenBCI_Cyton(port="COM3", daisy=False) | |
| 3 | EEG.Unicorn() | |
| 4 | EEG.LiveAmp(n_channels, sfreq) | |
| 5 | EEG.Mindwave() | |
| 6 | EEG.Muse() | |
| 7 | **Body_Motion = OpenSync.sensors.BodyMotion()** | Initialize and stream Kinect body motion data |
| 8 | Body_Motion.KinectBodyBasics() | |
| 9 | **Eye_Tracking = OpenSync.sensors.EyeTracking()** | Initialize, configure and stream Gazepoint data |
| 10 | Eye_Tracking.Gazepoint(biometrics=False) | |
| | **Functions for I/O Devices** | |
| 11 | keyboard = OpenSync.i_o.Keyboard(Name) | Create keyboard object for streaming data |
| | - keyboard.stream_keypress (PsychoPy_keyboard_object) | Stream the name of the pressed key |
| 12 | mouse = OpenSync.i_o.Mouse(Name, clickable_object=True, position=True, click_type=True) | Create mouse object for streaming data |
| | - mouse.stream_clicktype(PsychoPy_mouse_object) | Stream the type of click {left, middle, right} |
| | - mouse.stream_click(PsychoPy_mouse_object) | Stream the name of the clicked object |
| | - mouse.stream_pos(PsychoPy_mouse_object) | Stream the position of the mouse (in every frame) |
| 13 | joystick = OpenSync.i_o.Joystick(Name, keypress=True, pos=True) | Create joystick object for streaming data |
| | - joystick.stream_keypress(PsychoPy_joystick_object) | Stream the keypress event |
| | - joystick.stream_pos(PsychoPy_joystick_object) | Stream the position of the joystick |
| 14 | gamepad = OpenSync.i_o.Gamepad(Name) | Create gamepad object for streaming data |
| | - gamepad.stream_buttonpress(PsychoPy_gamepad_object) | |
| | **Function for Stimulus Markers** | |
| 15 | marker_obj = OpenSync.markers.marker(Name) | Create marker object for streaming markers |
| | - marker_obj.stream_marker(_marker) | Stream the stimulus marker (int/string) |
| | **Function for Recording Data on Disk** | |
| 16 | OpenSync.record_data("File Address") | Record all streams in a file with *.xdf* extension |

Table 3 shows the list of libraries and device APIs for different biosensors that we customized for the development of OpenSync.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 July 2021 doi:10.20944/preprints202107.0651.v1

Table 3. Customized libraries and device APIs used in OpenSync

| Type | Device | Source Language | Library and Functions |
| --- | --- | --- | --- |
| EEG | g.tec Unicorn | C++ | https://github.com/moeinrazavi/Unicorn_LSL |
| | BrainProducts LiveAmp | C++ | https://github.com/moeinrazavi/LiveAmp-LSL |
| | OpenBCI Cyton (+Daisy) | Python | https://github.com/moeinrazavi/OpenBCI-LSL |
| | NeuroSky Mindwave | --- | https://pypi.org/project/mindwavelsl/ |
| | Muse | --- | Link 1: BlueMuse_Application<br>Link 2: BlueMuse_Installation_Guide |
| GSR | eHealth Sensor v2.0 Arduino Shield | C, Python | https://github.com/moeinrazavi/eHealth-GSR-LSL |
| | Gazepoint Biometrics Device | Python | https://github.com/moeinrazavi/Gazepoint-Eyetracking-GSR-HeartRate--LSL |
| Eye-tracking | Gazepoint | Python | https://github.com/moeinrazavi/Gazepoint-Eyetracking-GSR-HeartRate--LSL |
| | Tobii for HTC VIVE | C++ | https://github.com/moeinrazavi/TobiiPro_SRanipal |
| Body Motion | Kinect | C++ | https://github.com/moeinrazavi/Kinect-BodyBasics-LSL |

## 3. Case Studies

### *3.1. Study 1: Using OpenSync in PsychoPy*

Figure 2 shows a code snippet of how to use OpenSync to synchronize multiple measures in PsychoPy.

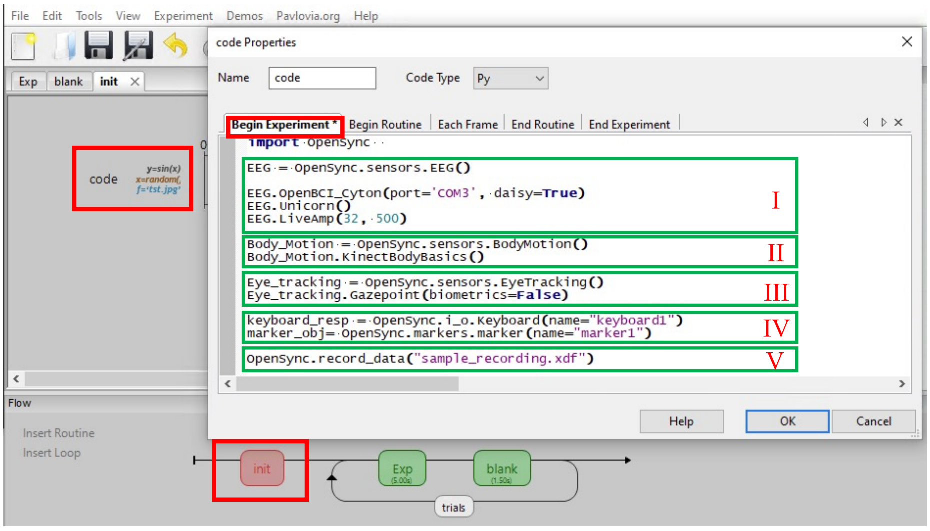



Figure 2.a. Code snippet for initialization of OpenSync; I) Setting up EEG sensors II) Setting up body motion sensors III) Setting up eye-tracking sensors IV) Define and initialization of I/O marker and stimulus marker objects V) Recording the data

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 July 2021 doi:10.20944/preprints202107.0651.v1

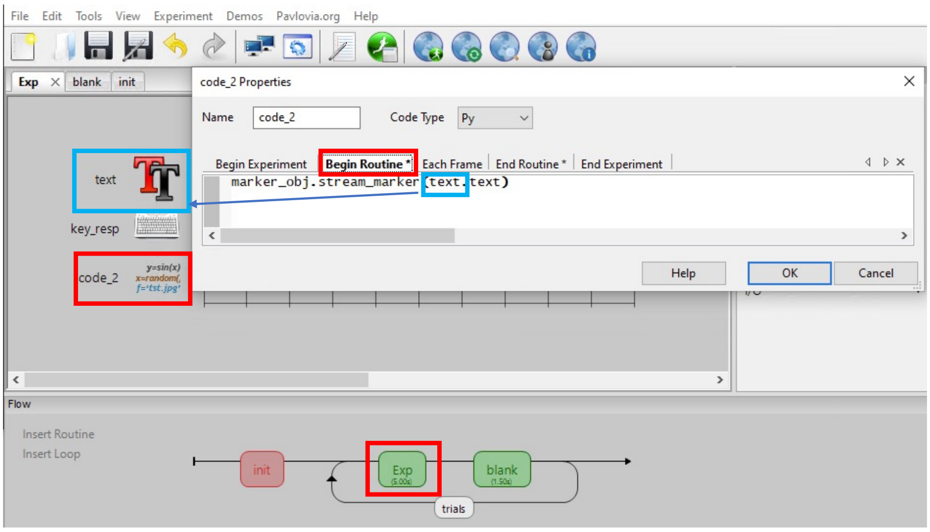



Figure 2.b. Code snippet for streaming stimulus markers

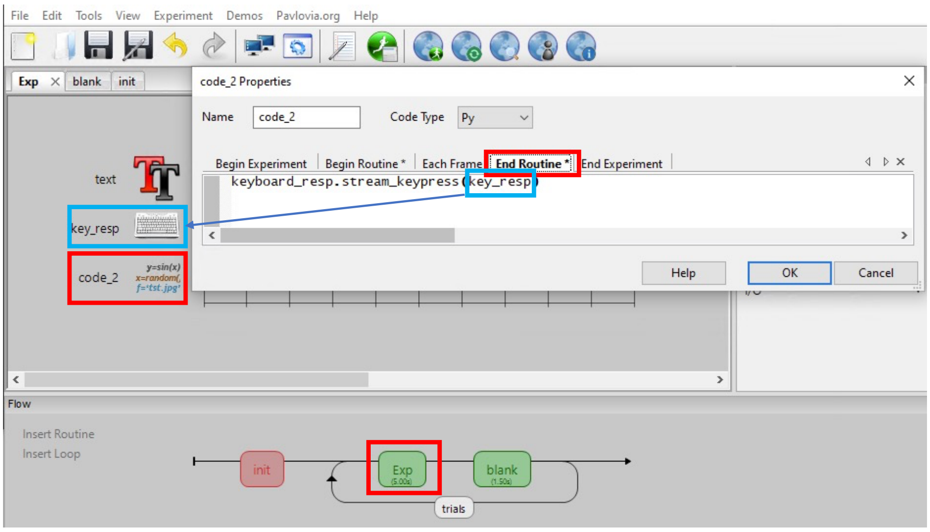
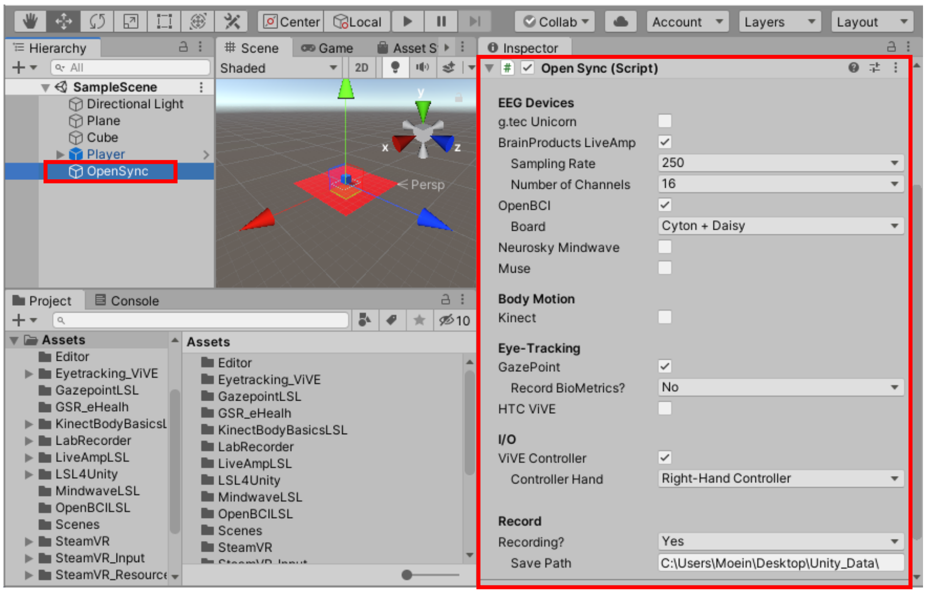



Figure 2.c. Code snippet for streaming user response (I/O) markers

As shown in Figure 2, data streams from multiple sources (e.g., EEG, eye-tracking, body motion, stimulus markers, and keyboard response markers) can be synchronized via OpenSync functions in PsychoPy. All the streams will be recorded in a single *.xdf* file on the disk by the function *OpenSync.record_data*(“file_name.xdf”).

*3.2. Study 2: Using OpenSync in Unity*

Figure 3. OpenSync in Unity




We have designed a Unity inspector that allows the user to stream data from multiple devices and synchronize them with each other through the GUI without writing code. The user can also set device configurations (e.g., in the EEG device BrainProducts LIVEAMP, they can set the Sampling Rate and Number of Channels). In order to record the streams, user only need to set the output path and data recording will be done automatically (Figure 3).

In order to use OpenSync Platform in Unity, OpenSync.cs and OpenSync_Editor.cs should be added to /Assets/ and /Assets/Editor/ folders, respectively. Then, OpenSync.cs should be attached to a GameObject in Unity scene. Figure 3. shows OpenSync attached to the GameObject named "OpenSync"; it contains some sample devices and measures. Other devices and measures can be easily added by following the same procedure in the OpenSync script.

## 4. Time Synchronization Test

In this section, we tested the OpenSync's ability for time-synchronization. To that end, we designed an experiment to compare the difference between nominal and average effective sampling times for a certain device in different conditions, i.e., when only data from that particular device are being recorded and when its data are being recorded together with data from several other devices. In our experiment we used g.tec Unicorn as the reference device and then added data streams from OpenBCI, Gazepoint, Kinect, Mouse and Keyboard devices.

*4.1 Experiment*

In the experiment, the device sends data to be saved on the disk and the average sample recording time is measured. Figure 4 shows the flow of this experiment.

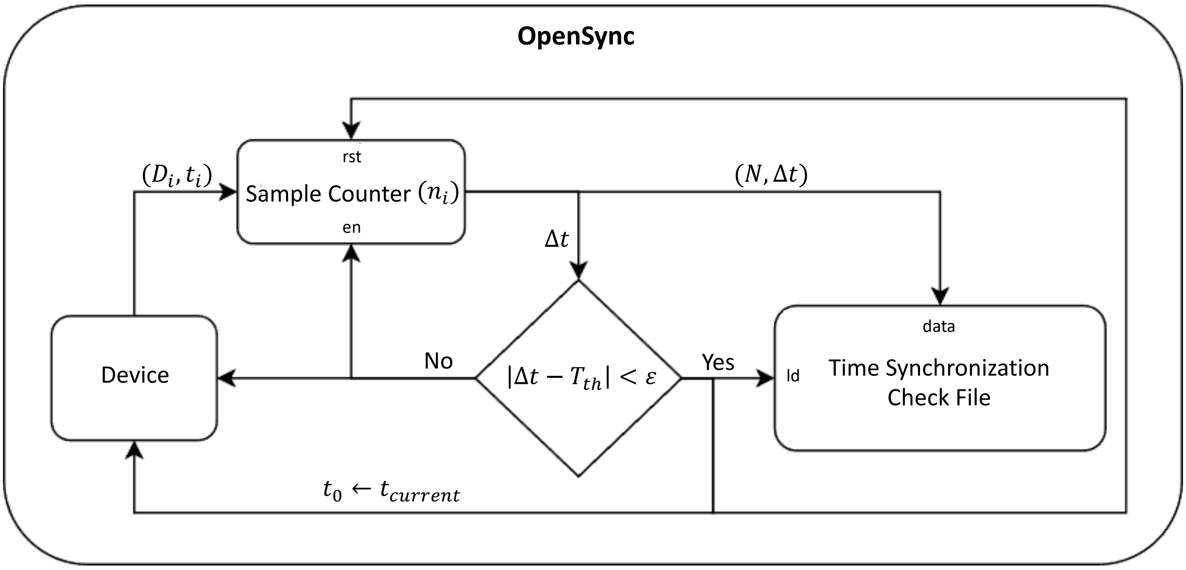



Figure 4. flowchart of the experiment

In this figure:

- $D_i$: $i^{th}$ data sample from the device
- $t_i$: timestamp at which the $i^{th}$ sample entered the data chunk in memory buffer
- $t_o$: timestamp at which the first sample entered the data chunk in memory buffer
- $T_{th}$: threshold time at which each data chunk is saved on the disk

Also, $N$ and $\Delta T$ can be obtained by:

$$N = \sum_{i=0}^{k} n_i,$$

$$\Delta T = t_k - t_0$$

The difference between the timestamps of two consecutive loops $(t_{i+1} - t_i)$ equals to the real sampling time plus the time lag induced by OpenSync for recording data for each sample. The average time lag $s$ per sample is derived by the following formula:

$$s = \left(\frac{\Delta T}{N}\right)_{|\Delta T - T_{th}| < \epsilon} - \frac{1}{SRate_{nom}}$$

where $\left(\frac{\Delta T}{N}\right)_{|\Delta T - T_{th}| < \epsilon}$ equals to the total time for recording a data chunk divided by the number of samples in that chunk when $|\Delta T - T_{th}| < \epsilon$. Also, $SRate_{nom}$ is the nominal sampling rate of the device. In OpenSync, $T_{th}$ is set on 500 ms for each chunk. Four different cases were included in the experiment:

- Case 1 – Unicorn only,
- Case 2 – Unicorn + OpenBCI,
- Case 3 – Unicorn + OpenBCI + Gazepoint,
- Case 4 – Unicorn + OpenBCI + Gazepoint + Kinect + Mouse + Keyboard

*4.2. Results*

In this section, we analyzed the delay caused by the OpenSync platform for synchronization. Figure 5 shows the distribution of the differences between Unicorn nominal sampling time (250 Hz $\equiv$ 400 μs) and the

 doi:10.20944/preprints202107.0651.v1

average real recorded time per sample using OpenSync platform for the four different cases. Table 4 shows the descriptive statistics of the experiment.

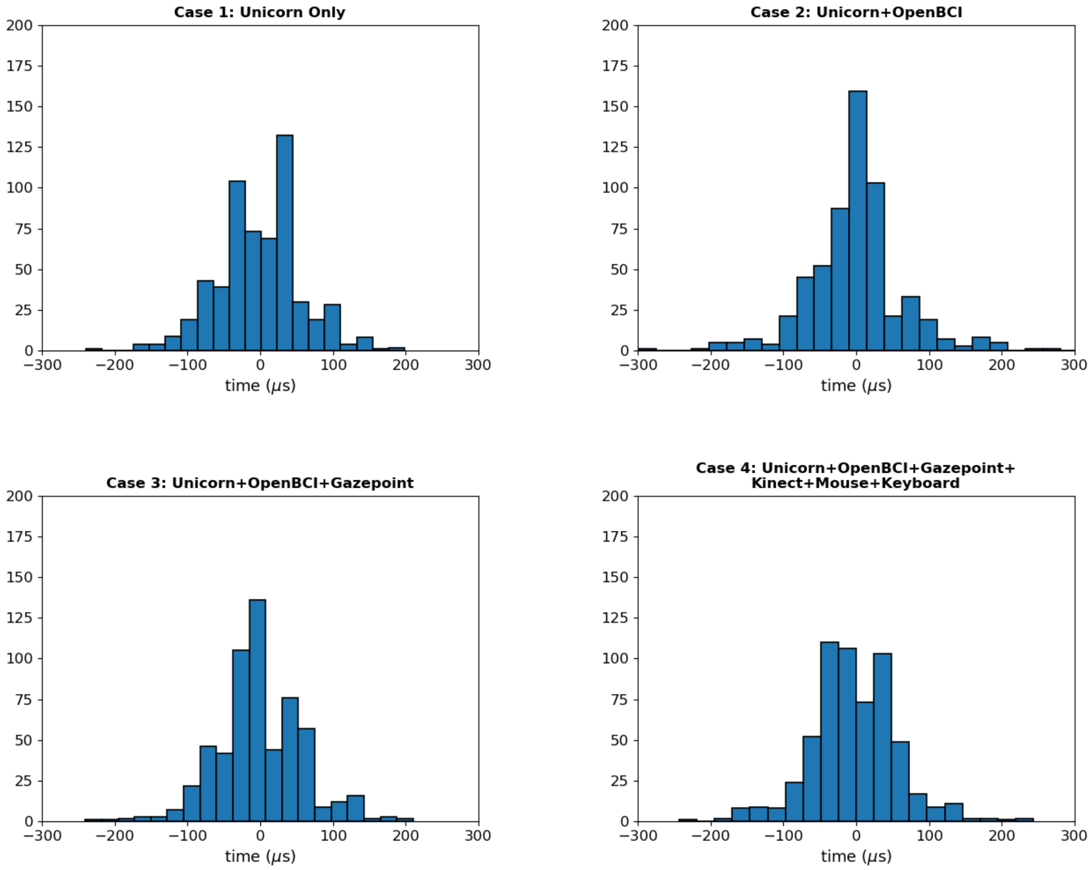



Figure 5. Results of running OpenSync in cases with different number of devices

Table 4. Descriptive statistics of the experiment

| Case | N_sample | Mean (μs) | SD | SE | Min (μs) | Max (μs) |
|---|---|---|---|---|---|---|
| 1 | 600 | -0.550 | 57.7 | 2.38 | -239.67 | 198.23 |
| 2 | 600 | 0.515 | 65.3 | 2.69 | -298.71 | 352.58 |
| 3 | 600 | -0.317 | 59.4 | 2.45 | -240.32 | 210.21 |
| 4 | 600 | -1.147 | 60.2 | 2.48 | -243.15 | 243.28 |

In this table, each sample is the average time lag of recording Unicorn data samples within each data chunk. Each chunk includes 500 *ms* of data recording and each condition contains 5 minutes (300 *s*) of data recording. Therefore, we have $n = 600$ samples for each condition. As shown in the table, the minimum and maximum time lags are in the microsecond range, which meets our resolution criteria.

To compare the time lag between the four conditions, first we used Levene's test to check the homogeneity of the variances between the distributions of these conditions [11]. The result showed that their variances are homogeneous, $F\,(3, 2396) = 0.04$, $p = 0.99$. Then, we applied Fisher's One-way ANOVA [12], [13]. No significant difference was found between the time lags in four conditions,

$F\,(3, 2396) = 0.08,\ p = 0.97,\ \eta = 0.6,\ power = 1.0$. This indicates that increasing the number of devices using OpenSync, does not affect the effective sampling rate of the devices.

## 5. Discussion

### *5.1. Comparison to Existing Tools*

In this section, we compare OpenSync with other platforms that are used for data acquisition in psychological and neuroscience experiments. Table 5 provides the detailed comparison between different platforms.

iMotions is a platform used for implementation of human behavior experiments by integrating multiple biometric sensors (EEG, eye-tracking, EDA, etc.) [6].

BioPac is a system that provides multiple EEG, eye-tracking, GSR, etc. devices along with data acquisition and analysis tools used for behavioral experiments [7].

OpenViBE and BCI2000 are platforms used for designing brain-computer interfaces that allow to connect different EEG devices along with several pre-configured and demo programs such as study/feedback paradigms [8], [9].

Naturalistic experimental design environment (NEDE) is a platform for Unity engine that integrates eye-tracking and EEG devices for experiments in 3D environment [14].

The Unified Suite for Experiments (USE) includes a set of hardware and software tools to integrate EEG and eye-tracking for behavioral neuroscience experiments [15].

Table 5. Comparison between different platforms

| | iMotions | BioPac | OpenViBE | BCI2000 | NEDE | USE | **OpenSync** |
|---|---|---|---|---|---|---|---|
| Free/Opensource | × | × | ✓ | ✓ | ✓ | ✓ | ✓ |
| Multi-Device Synchronization | ✓ | ✓ | × | × | ✓ | ✓ | ✓ |
| Platforms | × | × | C++ | C++ | JavaScript | C# | Python,C, C#, C++,Java, Matlab |
| Device Support | Limited | Limited | EEG Only | EEG Only | Limited | Limited | Unlimited |
| Comprehensive | ✓ | × | × | ✓ | × | × | ✓ |
| Easy Implementation | ✓ | ✓ | ✓ | ✓ | × | × | ✓ |
| No Extra Hardware | ✓ | × | ✓ | ✓ | × | × | ✓ |
| Automatic Record | × | × | × | × | × | × | ✓ |
| User Extendable | × | × | × | × | × | × | ✓ |
| Portable | × | × | × | × | × | × | ✓ |

### *5.2. Studies with multimodal measures*

A few studies tried to integrate multiple measures into the experimental psychology/neuroscience portal. Reeves et al. (2007) state the importance of using multiple measures in the goals of Augmented Cognition (AUGCOG) [16]; they discuss the combination of multiple measures together as a factor for the technologies that improve Cognitive State Assessment (CSA). Jimenez-Molina et al. (2018) showed that analyzing all measures of electrodermal activity (EDA), photoplethysmogram (PPG), EEG, temperature and pupil dilation at the same time, significantly

improves the classification accuracy in a web-browsing workload classification task, compared to using a single measure or a combination of some of them [17].

Born et al. (2019) used EEG, GSR and eye-tracking to predict task performance in a task load experiment; they found that low-beta frequency bands, pupil dilations and phasic components of GSR were correlated with task difficulty [18]. They also showed that the statistical results of analyzing EEG and GSR together were more reliable than analyzing them individually. Wang et al. (2014) used PsychoPy, EEG, and LSL for Brain-Computer Interface (BCI) stimulus presentations. They used these to synchronize the stimulus markers and EEG measurements [19]. Leontyev et al. combined user response time and mouse movement features with machine learning technics and found an improvement in the accuracy of predicting attention-deficit/hyperactivity disorder (ADHD) [20]–[22]. Yamauchi et al. combined behavioral measures and multiple mouse motion features to better predict people's emotions and cognitive conflict in computer tasks [23], [24]. Yamauchi et al. further demonstrated that people's emotional experiences change as their tactile sense (touching a plant) was augmented with visual sense ("seeing" their touch) in a multisensory interface system [25]. Razavi et al. provided a comprehensive tutorial on how use LSL for integrating multimodal experiments into psychological experiment design software, PsychoPy and Unity [26]. Chen et al. (2012) tried to identify possible correlations between increasing levels of cognitive demand and modalities from speech, digital pen, and freehand gesture to eye activity, galvanic skin response, and EEG [27]. Lazzeri et al. (2014) used physiological signals, eye gaze, video and audio acquisition to perform an integrated affective and behavioral analysis in Human-Robot Interaction (HRI) [28]; by acquiring synchronized data from multiple sources, they investigated how autism patients can interact with affective robots. Cornelio et al. reviewed the challenges and technological issues in creating multisensory experiments [29]. Charles and Nixon (2019) reviewed 58 articles on mental workload tasks. They found that physiological measurements such as ECG, respiration, GSR, blood pressure, EOG and EEG need to be triangulated because though they are sensitive to mental workload, no single measure satisfies to predict mental workload [30]. Lohani et al. (2019) suggest that analyzing multiple measures such as head movement together with physiological measures (e.g., EEG, heart rate, etc.) can be used for the drivers to detect cognitive states (e.g., distraction) [31]. Gibson et al. (2014) integrated questionnaires, qualitative methods, and physiological measures including ECG, respiration, electrodermal activity (EDA) and skin temperature to study activity settings in disabled youth [32]; they stated that using multiple measures reflects a better real-world setting of the youth experiences. Sciarini and Nicholson (2009) used EEG, eye blink, respiration, cardiovascular activity and speech measures in a workload task performance [33].

### *5.3. Summary and Future Work*

The ability to use multiple synchronized measures to distinguish factors affecting behavior and brain functionality is getting more attention. Multiple studies showed that using several measures (multimodal experiments) can improve the accuracy and the confidence for interpretation of the results. Synchronizing multiple measures with different sampling rates has various challenges that only few studies tried to integrate more than two measures. In this paper, we introduced OpenSync platform for integrating and synchronizing multiple measures. Once the experiment is created using this platform, it is straightforward and timesaving for the experimenter to run the experiment as every measure starts being recorded and saved on the disk automatically. This platform can be easily expanded and used for different purposes such as Brain-Computer Interface and Neurofeedback experiments [34]. In future, this platform can be expanded for multimodal-multisensory experiments that involve different human senses (e.g., tactile, hear, smell, taste, etc.). Applying OpenSync to different human senses will improve accuracy and is likely to advance our knowledge of human behavior [35]. For this purpose, the state-of-the-art deep learning models are powerful tools that have recently been used for combining and analyzing data from multiple sources together [36].

 doi:10.20944/preprints202107.0651.v1

## Appendix

*Lab Streaming Layer*

LSL is an application programming interface (API) that is available on open-source platforms (e.g., Python, C, C++, C#, Java, Octave, etc). It uses Transmission Control Protocol (TCP) for stream transport and User Datagram Protocol (UDP) for stream discovery and time synchronization. TCP is a connection-oriented protçthat guarantees errorless, reliable and ordered data streaming and it works with internet protocol (IP) for data streaming [37], [38] whereas UDP is a connectionless protocol [39]. The software development kit (SDK) of almost every consumer and research-grade device that supports one of the opensource platforms (e.g., Python, C, C++, C#, Java, Octave, etc.), can be customized to stream data via LSL.

*Installing OpenSync*

OpenSync can be installed for any Python environment using "*pip install OpenSync*". To install OpenSync on PsychoPy, set the current directory in the command line to PsychoPy Python directory (e.g., use "*cd C:\Program Files\PsychoPy3*" command) and then use "*python -m pip install OpenSync --user*".

To use OpenSync on Unity, download the files from *OpenSync_Unity* github repository (https://github.com/moeinrazavi/OpenSync_Unity) and copy them inside the */Assets/* folder in your Unity project. Then create a new GameObject in Unity and add the *OpenSync.cs* script to that GameObject. This will automatically add OpenSync GUI to Unity inspector.

*.xdf File Structure*

In order to open *.xdf* file in Python, first it is required to install *pyxdf* in python using pip in command line: "*pip install pyxdf*". The *.py* file in the link: pyxdf_example, is an example of opening *.xdf* files in Python. It is recommended to use *Spyder* (https://docs.spyder-ide.org/installation.html) as the Python platform to open the *.xdf* files, since the Variable Explorer panel in *Spyder* allows to track the variables. The fields of a *.xdf* file in Python are shown in shown Figure A1.

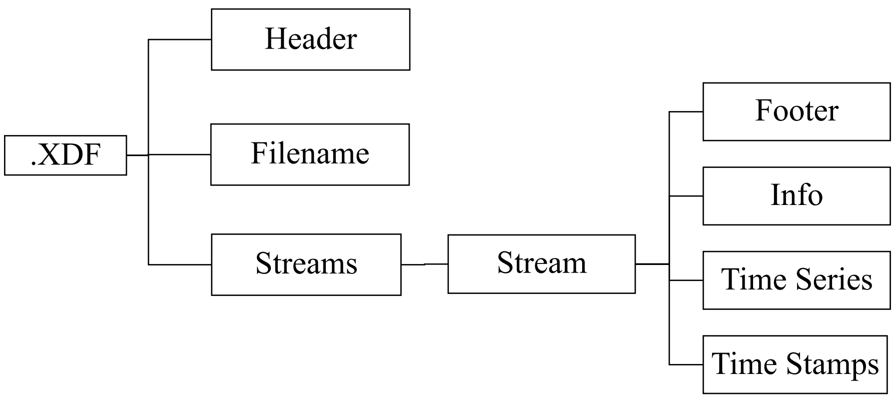



**Figure A1.** Fields of *.xdf* file

## References

[1] R. A. Stevenson *et al.*, “Identifying and Quantifying Multisensory Integration: A Tutorial Review,” *Brain Topogr.*, vol. 27, no. 6, pp. 707–730, 2014.

[2] T. U. Otto, B. Dassy, and P. Mamassian, “Principles of multisensory behavior,” *J. Neurosci.*, vol. 33, no. 17, pp. 7463–7474, 2013.

[3] R. Höchenberger, N. A. Busch, and K. Ohla, “Nonlinear response speedup in bimodal visual-olfactory object identification,” *Front. Psychol.*, vol. 6, no. September, pp. 1–11, 2015.

[4] H. D. Critchley *et al.*, “Human cingulate cortex and autonomic control: Converging neuroimaging and clinical evidence,” *Brain*, vol. 126, no. 10, pp. 2139–2152, 2003.

[5] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, “On combining classifiers,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, 1998.

[6] “iMotions.” [Online]. Available: https://imotions.com/. [Accessed: 29-May-2021].

[7] “Data Acquisition, Loggers, Amplifiers, Transducers, Electrodes | BIOPAC.” [Online]. Available: https://www.biopac.com/. [Accessed: 29-May-2021].

[8] Y. Renard *et al.*, “OpenViBE: An open-source software platform to design, test, and use brain-computer interfaces in real and virtual environments,” *Presence Teleoperators Virtual Environ.*, vol. 19, no. 1, pp. 35–53, 2010.

[9] G. Schalk *et al.*, “BCI2000: A General-Purpose Brain-Computer Interface (BCI) System,” *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, 2004.

[10] “Introduction — Labstreaminglayer 1.13 documentation.” [Online]. Available: https://labstreaminglayer.readthedocs.io/info/intro.html. [Accessed: 31-May-2021].

[11] H. Levene and others, “Contributions to probability and statistics,” *Essays Honor Harold Hotell.*, pp. 278–292, 1960.

[12] R. A. Fisher, “XV.—The Correlation between Relatives on the Supposition of Mendelian Inheritance,” *Trans. R. Soc. Edinburgh*, vol. 52, no. 2, pp. 399–433, 1919.

[13] R. A. Fisher, “On the’probable error’of a coefficient of correlation deduced from a small sample,” *Metron*, vol. 1, pp. 1–32, 1921.

[14] D. C. Jangraw, A. Johri, M. Gribetz, and P. Sajda, “NEDE: An open-source scripting suite for developing experiments in 3D virtual environments,” *J. Neurosci. Methods*, vol. 235, pp. 245–251, 2014.

[15] M. R. Watson, B. Voloh, C. Thomas, A. Hasan, and T. Womelsdorf, “USE: An integrative suite for temporally-precise psychophysical experiments in virtual environments for human, nonhuman, and artificially intelligent agents,” *J. Neurosci. Methods*, vol. 326, no. June, 2019.

[16] L. M. Reeves, D. D. Schmorrow, and K. M. Stanney, “Augmented cognition and cognitive state assessment technology - Near-term, mid-term, and long-term research objectives,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4565 LNAI, pp. 220–228, 2007.

[17] A. Jimenez-Molina, C. Retamal, and H. Lira, “Using psychophysiological sensors to assess mental workload during web browsing,” *Sensors (Switzerland)*, vol. 18, no. 2, pp. 1–26, 2018.

[18] J. Born, B. R. N. Ramachandran, S. A. Romero Pinto, S. Winkler, and R. Ratnam, “Multimodal study of the effects of varying task load utilizing EEG, GSR and eye-tracking,” *bioRxiv*, no. 1998, p. 798496, 2019.

[19] Z. Wang, G. Healy, A. F. Smeaton, and T. E. Ward, “An investigation of triggering approaches for the rapid serial visual presentation paradigm in brain computer interfacing,” *2016 27th Irish Signals Syst. Conf. ISSC 2016*, pp. 1–6, 2016.

[20] A. Leontyev, T. Yamauchi, and M. Razavi, “Machine Learning Stop Signal Test (ML-SST): ML-based

Mouse Tracking Enhances Adult ADHD Diagnosis," in *2019 8th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos, ACIIW 2019*, 2019, pp. 248–252.

[21] A. Leontyev, S. Sun, M. Wolfe, and T. Yamauchi, "Augmented Go/No-Go Task: Mouse Cursor Motion Measures Improve ADHD Symptom Assessment in Healthy College Students," *Front. Psychol.*, vol. 9, no. APR, p. 496, Apr. 2018.

[22] A. Leontyev and T. Yamauchi, "Mouse movement measures enhance the stop-signal task in adult ADHD assessment," *PLoS One*, vol. 14, no. 11, p. e0225437, 2019.

[23] T. Yamauchi and K. Xiao, "Reading Emotion From Mouse Cursor Motions: Affective Computing Approach," *Cogn. Sci.*, vol. 42, no. 3, pp. 771–819, Apr. 2018.

[24] T. Yamauchi, A. Leontyev, and M. Razavi, "Assessing Emotion by Mouse-cursor Tracking: Theoretical and Empirical Rationales," in *2019 8th International Conference on Affective Computing and Intelligent Interaction, ACII 2019*, 2019, pp. 89–95.

[25] T. Yamauchi, J. Seo, and A. Sungkajun, "Interactive Plants: Multisensory Visual-Tactile Interaction Enhances Emotional Experience," *Mathematics*, vol. 6, no. 11, p. 225, Oct. 2018.

[26] M. Razavi, T. Yamauchi, V. Janfaza, A. Leontyev, S. Longmire-Monford, and J. Orr, "Multimodal-Multisensory Experiments." Preprints, 12-Oct-2020.

[27] F. Chen *et al.*, "Multimodal behavior and interaction as indicators of cognitive load," *ACM Trans. Interact. Intell. Syst.*, vol. 2, no. 4, 2012.

[28] N. Lazzeri, D. Mazzei, and D. De Rossi, "Development and Testing of a Multimodal Acquisition Platform for Human-Robot Interaction Affective Studies," *J. Human-Robot Interact.*, vol. 3, no. 2, p. 1, 2014.

[29] P. Cornelio, C. Velasco, and M. Obrist, "Multisensory Integration as per Technological Advances: A Review," *Front. Neurosci.*, vol. 15, Jun. 2021.

[30] R. L. Charles and J. Nixon, "Measuring mental workload using physiological measures: A systematic review," *Applied Ergonomics*, vol. 74. Elsevier Ltd, pp. 221–232, 01-Jan-2019.

[31] M. Lohani, B. R. Payne, and D. L. Strayer, "A review of psychophysiological measures to assess cognitive states in real-world driving," *Front. Hum. Neurosci.*, vol. 13, no. March, pp. 1–27, 2019.

[32] B. E. Gibson *et al.*, "A multi-method approach to studying activity setting participation: Integrating standardized questionnaires, qualitative methods and physiological measures," *Disabil. Rehabil.*, vol. 36, no. 19, pp. 1652–1660, 2014.

[33] L. W. Sciarini and D. Nicholson, "Assessing cognitive state with multiple physiological measures: A modular approach," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5638 LNAI, pp. 533–542, 2009.

[34] R. Abiri, S. Borhani, E. W. Sellers, Y. Jiang, and X. Zhao, "A comprehensive review of EEG-based brain-computer interface paradigms," *Journal of Neural Engineering*, vol. 16, no. 1. Institute of Physics Publishing, p. 011001, 01-Feb-2019.

[35] E. Marsja, J. E. Marsh, P. Hansson, and G. Neely, "Examining the role of spatial changes in bimodal and uni-modal to-be-ignored stimuli and how they affect short-term memory processes," *Front. Psychol.*, vol. 10, no. MAR, pp. 1–8, 2019.

[36] J. Gao, P. Li, Z. Chen, and J. Zhang, "A survey on deep learning for multimodal data fusion," *Neural Comput.*, vol. 32, no. 5, pp. 829–864, 2020.

[37] J. Postel, "DOD standard transmission control protocol," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 10, no. 4, pp. 52–132, 1980.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 July 2021 doi:10.20944/preprints202107.0651.v1

[38] W. R. Stevens, *TCP/IP Illustrated (Vol. 1): The Protocols*. USA: Addison-Wesley Longman Publishing Co., Inc., 1993.

[39] P. Loshin, "User Datagram Protocol," *TCP/IP Clear. Explain.*, no. August, pp. 341–349, 2003.